Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Robust Low-Rank Tensor Factorisations: New Algorithms and Extensive Comparisons

Author:
Mehdi Bahri

Supervisor:
Stefanos Zafeiriou

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing of Imperial College London

September 2016

Abstract

Sparse methods have attracted strong interest over the past decade. The success of robust decompositions that leverage signal incoherence and sparsity to allow for exact reconstruction under more flexible conditions than those given by fixed-rate sampling have sprouted active research in statistics, signal processing, and artificial intelligence. In this thesis, we review the recent advances in robust tensor factorisations and propose new algorithms based on a two-dimensional decompositions of the tensor frontal slices. We then present an extensive experimental assessment of our methods and of the state of the art on a variety of computer vision benchmarks, and demonstrate the superior performance of our models compared to existing algorithms. Finally, we set the basis for a Bayesian treatment of our methods and provide a software package for further research and experiments.

Acknowledgments

I would like to thank my supervisor Dr Stefanos Zafeiriou, and Dr Yannis Panagakis for their support and guidance throughout this project. Working with them has been nothing short of a privilege.

Many thanks go to the authors who provided code, either on their websites or on request.

Finally, I would like to thank my family, my friends, and my girlfriend L. D. for their love and encouragement.

Contents

1 Introduction **1**
- 1.1 The low-rank data assumption . . . 1
- 1.2 Recovering signals from noisy measurements . . . 2
- 1.3 Compressed Sensing and inverse problems . . . 2
- 1.4 Why tensors ? . . . 3
- 1.5 Contributions . . . 3

2 Mathematical background and preliminaries **4**
- 2.1 General linear algebra . . . 4
  - 2.1.1 Matrix rank . . . 4
  - 2.1.2 Singular Value Decomposition . . . 5
  - 2.1.3 Matrix norms . . . 5
  - 2.1.4 Vectorisation operator and Kronecker product . . . 8
- 2.2 Multilinear algebra . . . 9
  - 2.2.1 Notations . . . 9
  - 2.2.2 Tensor basics . . . 10
  - 2.2.3 The CP and Tucker decompositions . . . 12
  - 2.2.4 Tensor norms and SVDs . . . 13
  - 2.2.5 Tensor ranks and their convex surrogates . . . 16
- 2.3 Convex optimisation . . . 18
  - 2.3.1 Proximal operators . . . 18
  - 2.3.2 Augmented Lagrangian Function . . . 19
  - 2.3.3 Method of Alternating Direction of Multipliers and variable splitting . . . 20
- 2.4 Multivariate statistics and Bayesian inference . . . 20
  - 2.4.1 Some matrix-valued probability distributions . . . 20
  - 2.4.2 Kullback-Leibler divergence . . . 21
  - 2.4.3 Mean-field approximation in directed graphical models . . . 21
- 2.5 Image quality measures . . . 22
  - 2.5.1 Measures of pixel-wise similarity . . . 22
  - 2.5.2 Measures of structural similarity . . . 23
  - 2.5.3 Spectral Angle Mapper . . . 24

3 Low-rank modelling on matrices and tensors: a literature review **25**
- 3.1 Low-rank modelling on matrices . . . 25
  - 3.1.1 Optimisation-Based Matrix decompositions . . . 25
  - 3.1.2 Probabilistic Matrix Decompositions . . . 27
- 3.2 Low-rank modelling on tensors . . . 30
  - 3.2.1 Matricisation-based methods . . . 30

3.2.2 Robust Factorisations . . . 32
3.2.3 Frontal slices . . . 34
3.2.4 Other optimisation-based approaches . . . 36
3.2.5 Bayesian methods . . . 38

4 Novel algorithms for Robust 2D PCA **40**
4.1 Introduction . . . 40
4.2 Generic form of the optimisation problem . . . 40
4.3 Non-orthogonal Robust 2D PCA . . . 41
4.3.1 Optimisation problem . . . 41
4.3.2 Augmented Lagrangian . . . 41
4.3.3 Update equations and algorithm . . . 42
4.4 Sparse Non-Orthogonal Robust 2D PCA . . . 44
4.4.1 Optimisation problem and solution by splitting . . . 45
4.4.2 Update equations and algorithm . . . 45
4.5 Group Lasso-penalised bases . . . 46
4.5.1 Optimisation problem . . . 47
4.5.2 Solution with an inexact proximal gradient step . . . 47
4.5.3 Solution using ADMM and splitting . . . 47
4.6 Iterative robust estimation of the mean . . . 48
4.7 Model validation on synthetic data . . . 50
4.7.1 Experimental setup . . . 51
4.7.2 Sample run and automated tuning . . . 52
4.7.3 Sparsity of the solution and rank recovery . . . 56
4.8 Discussion . . . 56
4.8.1 Convergence . . . 57
4.8.2 Variable initialisation . . . 59
4.8.3 Complexity analysis and scalability . . . 60
4.8.4 Implementation details . . . 61
4.8.5 Relation to sparse dictionary learning . . . 63
4.9 Summary . . . 64

5 Comparison with the state of the art **65**
5.1 Experimental setup . . . 65
5.1.1 Algorithms compared . . . 65
5.1.2 Parameters tuning . . . 67
5.2 Background subtraction . . . 68
5.2.1 The highway dataset . . . 69
5.2.2 The airport hall dataset . . . 75
5.3 Denoising of monochromatic face images . . . 79
5.3.1 Salt & pepper noise . . . 79
5.3.2 Random patch occlusion . . . 89
5.4 Colour image denoising . . . 100

6 Basis for a Bayesian treatment of Robust 2D PCA **107**
6.1 Choice of priors . . . 107
6.1.1 Sparse error . . . 107
6.1.2 Latent core tensor . . . 108
6.1.3 Left and right bases . . . 108

6.1.4 Distribution of each $\boldsymbol{\gamma}_i$ . . . 108
6.1.5 White noise and distribution of $\mathbf{Y}_n$ . . . 109
6.2 Joint distribution and graphical model . . . 109
6.3 Solution with variational inference . . . 109
6.3.1 Logarithms of the distributions . . . 109
6.3.2 Variational updates . . . 110
6.4 Required intermediary computations . . . 115
6.4.1 Expectation of $\mathbf{w}_{ni}^{\mathsf{T}}\mathbf{w}_{ni}$ . . . 115
6.4.2 Expectation of $||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}} - \mathbf{E}_n||_{\mathrm{F}}^2$ . . . 116
6.5 Performance . . . 117

7 Conclusion **118**
7.1 Future work and extensions . . . 119
7.1.1 Sparse dictionary learning . . . 119
7.1.2 Applicability to data compression . . . 119
7.1.3 Visual quality assessment by a human jury . . . 119
7.1.4 Orthogonal bases . . . 119
7.1.5 Non-visual data . . . 119
7.1.6 Extension to other norms and missing values . . . 120
7.1.7 Bayesian model with time correlation . . . 120
7.1.8 Theoretical analysis . . . 120

A Complexity Analysis **121**

B Best parameters per algorithm **125**

C Parameter tuning: experimental plots **134**

List of Tables **143**

List of Figures **145**

Bibliography **145**

Chapter 1

Introduction

In this section we present the theoretical and practical considerations that motivate the present thesis. We then give an overview of our contribution to the field.

1.1 The low-rank data assumption

Multidimensional data are common in engineering and scientific applications, and a high number of dimensions allows us to embed more information into the data. Most data, whether they represent the state of a system, images, documents, sound, or other, can be mathematically represented as points in n-dimensional vector spaces over a given scalar field - usually real numbers as many real-world data are real-valued.

Several problems arise with high dimensional data, and especially in the common case of limited sample size. The *curse of dimensionality* refers to the difficulty in finding neighbouring points in high dimensional spaces - or, to put it differently, to the *sparsity* of high dimensional spaces - and has detrimental consequences on the performance and speed of learning algorithms. Standard Markov Chain Monte Carlo algorithms generally exhibit a random walk behaviour, and therefore have very slow convergence in high dimensional spaces. Algorithms that rely on computing distances between data points are hindered by the fact that *similar* data points can appear very *distant* due to an artificial increase in the dimension of the space in which they reside.

The underlying assumption here is that many usual data can be accurately represented with fewer dimensions than when observed. Mathematically, the geometric interpretation is that the true data reside on a low-dimensional manifold embedded in the high-dimensional space. Empirical justifications of such an assumption are diverse. The observed data can be *sampled* from that manifold with the addition of noise, or can carry redundant information due to correlations between the dimensions. Let $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_N]$ be the matrix whose columns are the observed data points represented as n-dimensional vectors. *Statistical correlation* between the observations translates as *colinearity* between the points, hence making the matrix **low-rank** - as opposed to full-rank - and therefore reducing the **dimension** of the space $\mathrm{Span}(\mathbf{X})$ spanned by the columns of $\mathbf{X}$. We can also seek the low-dimensional data that *best summarise* - or *best explain* - the observed data.

Such an endeavour dates back as early as 1901 with the famous, and still widely popular, *Principal Component Analysis* (Pearson [78]), that aims at finding an orthogonal projection of the data onto a linear subspace that best preserves the observe diversity of the data (*i.e,* the statistical *variance*). This and all the subsequent work up to this day form the thriving field of **dimensionality reduction** techniques and *low-rank modelling*.

1.2 Recovering signals from noisy measurements

One of the earliest and yet most important results of signal processing is the Nyquist-Shannon Sampling theorem (Nyquist [74] (1928), Shannon [88] (1949)), which provides a **sufficient condition** for perfect recovery of signals of bounded bandwidth under the addition of noise. If a signal contains no frequencies higher than B Hertz, then it is possible to unambiguously recover the true signal from a sampled signal provided that the sampling rate is strictly greater than $2B$.

The classical sampling theorem is about **fixed-rate sampling** and its conditions are **not necessary**: perfect recovery can still be achieved at lower fixed sampling rates. This gave rise to the recent field of **Compressed Sensing**, which leverages the *sparsity* and *incoherence* of signals to allow for perfect recovery of noisy signals under milder conditions by formulating suitable optimisation problems.

1.3 Compressed Sensing and inverse problems

Inverse problems refer to the process of computing the causes of a phenomenon from the observations. As an example, image denoising is an inverse problem where the goal is to compute the inverse mapping of an operator $\boldsymbol{\mathcal{A}}$ that adds noise to the image. More formally, consider a noisy image represented as a vector $\mathbf{y}$, obtained from a *ground-truth* image $\mathbf{x}$ with the addition of noise:

$$\boldsymbol{\mathcal{A}}\mathbf{x} = \mathbf{y} \tag{1.1}$$

Can we recover $\mathbf{x}$ from the observed $\mathbf{y}$? If $\boldsymbol{\mathcal{A}}$ is linear, the problem is known as a *linear inverse problem*, and we call $\boldsymbol{\mathcal{A}}$ the *sensing matrix*.

Such problem is generally *underdetermined* and therefore *ill-posed*. Even if this inverse mapping exists, computing it numerically remains challenging due to numerical instability. In the simple case of an invertible linear operator $\boldsymbol{\mathcal{A}}$, computing $\mathbf{x} = \boldsymbol{\mathcal{A}}^{-1}\mathbf{y}$ is generally numerically difficult, even though mathematically sufficient. A common solution to this is to stabilise the computations by solving optimisation problems with the addition of *regularisation*. The signal processing and statistical literature is rich on the subject: successful examples include *Tikhonov* or ℓ_2-norm regularisation introduced by Tikhonov in 1943 and often found in the statistical literature under the name of *ridge regression* (see Tikhonov et al. [98], Hastie et al. [48] for a modern treatment of the matter) that adds a squared ℓ_2-norm penalty on the solution to stabilise its norm. Another regularisation approach that will be central to this thesis is the *Lasso* (Tibshirani [97], Hastie et al. [48]) or ℓ_1-norm regularisation which uses the ℓ_1 norm to promote sparsity in the solution.

Inverse problems and Compressed Sensing are tightly interconnected. The key insight is that sparse signals carry redundant information and therefore require fewer measurements to be reconstructed exactly than given by the classical fixed-rate sampling theory. Candès and Tao [22, 19, 20] derived the *Restricted Isometry Property (RIP)*; a sufficient condition of incoherence for sparse signals is that the sensing matrix be sufficiently close to an isometry: $(1 - \delta_s)||\mathbf{x}||_{\ell_2}^2 \leq ||\boldsymbol{\mathcal{A}}\mathbf{x}||_{\ell_2}^2 \leq (1 + \delta_s)||\mathbf{x}||_{\ell_2}^2$ for $\mathbf{x}$ s-sparse (*i.e,* has at most s non-zero coefficients).

A recent and ground-breaking development in the Compressed Sensing literature concerns the problem known as *Robust PCA*, which aims at decomposing a matrix into the sum of a *low-rank* component, and *sparse outliers* of arbitrarily large magnitude. Sparsity is indeed the key here, as a low-rank assumption on a matrix $\mathbf{A}$ is equivalent to sparsity in the spectrum of $\mathbf{A}$. More generally, Compressed Sensing requires that the signals be sparse *in some domain*. Under the *RIP*, exact recovery can be obtained with overwhelming probability via numerical optimisation.

We give a more thorough description of the Robust PCA problem in section 3.1.1 and refer the reader to Tarantola [95] (2004) for more information on inverse problems.

1.4 Why tensors ?

As we mentioned in the beginning of this introduction, multidimensional data are ubiquitous because *reality is* multidimensional in essence; so much that many usual data exhibit a *higher-order structure* that is lost if the data are represented in vector or matrix form:

- Video sequences are made from *2D frames* that appear in *chronological order*, therefore having two spatial dimensions and one temporal dimension. The time dimension is important as successive frames are likely to be highly correlated. Representing a sequence as a vector or as a large block-matrix loses the spatial correlation information, the time correlation information, or both.
- RGB colour images are tri-dimensional, and can be represented by three two-dimensional components: the red, green, and blue components.
- Hyperspectral images such as those obtained from satellites contain information carried by light outside of the visible portion of the electromagnetic spectrum and embed many spectral bands.
- Face images of a single person under different lighting conditions can be represented as 3D data.

Higher-order arrays, mathematically known as tensors, are better representations for such data. The jump from matrix and vector algorithms to tensor models for low-rank modelling is however non-trivial, as will be explained in section 2.2.

1.5 Contributions

In this thesis, we provide the following contributions to the field of low-rank modelling on tensors:

- We summarise in a single coherent discussion the mathematical approaches to rank minimisation on tensors.
- We present a review of the most recent methods for robust tensor factorisation, both deterministic and Bayesian, and compare the performance of a selected subset of algorithms on a range of computer vision benchmarks.
- We present a class of deterministic models for image restoration and solve four special cases via convex optimisation; we validate our models experimentally on synthetic data, discuss their convergence, complexity and scalability, provide insight on distributed implementations for large problems, and make explicit links between our models and sparse dictionary learning.
- We provide an efficient parallel implementation for each of the four optimisation-based models and show their performance on real-world data; we demonstrate competitive to superior performance over the current state of the art.
- We set the basis for a Bayesian treatment of our models, show how to perform inference via Variational Bayes, and provide an implementation of the algorithm.
- We make publicly available a Matlab package with the algorithms, the benchmarks, and several tools for easy evaluation of the performance of other Tensor methods.

Chapter 2

Mathematical background and preliminaries

The diversity of approaches to robust low-rank modelling on tensors makes it a mathematically-rich field that draws mainly on linear and multilinear algebra, statistics, and optimisation. This chapter is dedicated to providing the reader with the basic knowledge required to understand the methods we develop in this thesis, as well as we the ones reviewed in Chapter 3. We include some of the recent developments in tensor algebra, and grant particular attention to giving a clear picture of the various definitions of the tensor rank and of their convex surrogates. Finally, we present the quantitative metrics we rely on in the experimental assessments of Section 4.7 and Chapter 5.

2.1 General linear algebra

To make the text of this thesis self-contained, we first review some knowledge of linear algebra we use in our models. We chose to focus on the case of matrices and vectors of *real numbers*, and hence do not aim at presenting the most generic results, but rather to give the reader an understanding of the tools we manipulate.

2.1.1 Matrix rank

The rank of a matrix can be defined in several equivalent ways, we hereby give a simple definition as well as important geometric results.

Definition 2.1.1. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ an $m \times n$ matrix of real numbers. The **rank** of $\mathbf{A}$ is defined equivalently as:

- The number of linearly independent columns of $\mathbf{A}$
- The number of linearly independent rows of $\mathbf{A}$

And is denoted $\mathrm{rank}(\mathbf{A})$.

The following property is immediate:

Property 2.1.1. *Let* $\mathbf{A} \in \mathbb{R}^{m \times n}$*, then* $\mathrm{rank}(\mathbf{A}) \leq \min(m, n)$.

A geometric interpretation of the matrix rank is given below.

Property 2.1.2. *Let* $\mathbf{A} \in \mathbb{R}^{m \times n}$*, then* $\mathrm{rank}(\mathbf{A}) = \dim(\mathrm{Span}(\mathbf{A}))$.
Where $\mathrm{Span}(\mathbf{A})$ *denotes the* $\mathbb{R}$*-vector space spanned by the columns of* $\mathbf{A}$.

We also remind the reader of the link between matrix rank and matrix invertibility as this plays an important role in the resolution of linear systems, but first define the concept of full-rank matrix.

Definition 2.1.2. Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ a real matrix. $\mathbf{A}$ is said to be *full-rank* if and only if $\mathrm{rank}(\mathbf{A}) = \min(m, n)$.

Definition 2.1.3. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ a **square** real matrix. $\mathbf{A}$ is said to be *invertible* if and only if there exists a matrix $\mathbf{A}^{-1} \in \mathbb{R}^{n\times n}$ such that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I}_n$.

Property 2.1.3. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$, a **square** real matrix, then $\mathbf{A}$ is invertible if and only if $\mathbf{A}$ is full-rank.*

2.1.2 Singular Value Decomposition

Matrix decompositions are critical to numerical linear algebra, statistics, and optimisation. Many decompositions exist that are all important for different problems, such as the *QR* decomposition, the *Cholesky* decomposition, or the *Schur* decomposition - some of which we will refer to in the present document. However, the most important decomposition for low-rank modelling is without a doubt the *Singular Value Decomposition*, given by theorem 2.1.5 for matrices over the field of real numbers.

Let us first remind the reader of the definition and of some properties of orthogonal matrices. Orthogonal matrices are matrices for which the set columns is an orthonormal family of vectors as expressed in definition 2.1.4.

Definition 2.1.4. A real matrix $\mathbf{M}$ is said to be **orthogonal** or **unitary** if $\mathbf{M}\mathbf{M}^\mathsf{T} = \mathbf{M}^\mathsf{T}\mathbf{M} = \mathbf{I}$.

Since they are invertible, orthogonal matrices are full rank.

Geometrically, orthogonal matrices represent orthogonal bases (and are therefore change of basis matrices) where each direction is represented by a unit-length vector. Equivalently, orthogonal matrices represent unitary transformations, *i.e,* they preserve distances and as such are **isometries** of the space. From an algebraic point of view, the set of orthogonal matrices is a group, called the *orthogonal group* and is usually written $\mathcal{O}(n)$ in dimension n. One property of orthogonal matrices is that their determinant is always 1 or -1. The sub-set of orthogonal matrices with determinant 1 form a sub-group of the orthogonal group called the *special orthogonal group* and represent *rotations* of the space.

The Singular Value Decomposition describes a pair of orthogonal matrices such that the decomposed matrix $\mathbf{X}$ is equivalent to a diagonal matrix, whose coefficients are known as the *singular values* of $\mathbf{X}$.

Theorem 2.1.5. *Let $X \in \mathbb{R}^{n\times m}$ of rank r, there exists $\mathbf{U} \in \mathbb{R}^{n\times n}$, $\mathbf{V} \in \mathbb{R}^{m\times m}$ and $(s_1, \ldots, s_r) \in \mathbb{R}_+^r$ such that:*

$$X = \sum_{i=1}^{r} s_i u_i v_i^\mathsf{T} = \mathbf{U}S\mathbf{V}^\mathsf{T}$$

*With $\mathbf{U}$, $\mathbf{V}$ **orthogonal**, $\mathbf{S} = \begin{pmatrix} \mathbf{S}_r & 0 \\ 0 & 0 \end{pmatrix}$ and $\mathbf{S}_r = diag(s_1, \ldots, s_r)$. $\mathbf{S}$ is the matrix of the **singular values** of $\mathbf{X}$.*

When $\mathbf{X}$ is a square diagonalisable matrix, the SVD coincides with the Eigen decomposition.

2.1.3 Matrix norms

We now define some general forms of matrix norms as well as special cases of importance to our work.

Matrix operator norm

We first give a very general definition of the operator norm for completeness, property 2.1.4 describes the special case of real-valued matrices of finite dimensions.

Definition 2.1.6. Let f be an operator between two Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ equipped respectively with norms $||.||_1$ and $||.||_2$; then the operator norm of f induced by $||.||_1$ and $||.||_2$ is defined as:

$$|||f||| = \sup_{||\mathbf{v}||_1=1} ||f(\mathbf{v})||_2 = \sup_{||\mathbf{v}||_1\neq 0} \frac{||f(\mathbf{v})||_2}{||\mathbf{v}||_1}$$

It is the supremum of $||f(\mathbf{v})||_2$ on the unit ball of $\mathcal{H}_1$.

Property 2.1.4. *If $\mathcal{H}_1 = \mathbb{R}^m,\ \mathcal{H}_2 = \mathbb{R}^n$ with the Euclidean norm, then $|||f|||$ is the largest singular value of f.*

Proof. Let f a linear map from $\mathbb{R}^m$ to $\mathbb{R}^n$ represented by the matrix $\mathbf{X}$, and $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}$ its SVD.

Let $\mathbf{y}$ on the unit ball of $\mathbb{R}^m$, then:

$$\sup ||\mathbf{X}\mathbf{y}|| = \sup ||\mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}\mathbf{y}|| = \sup ||\mathbf{S}\mathbf{y}||$$

By definition of an isometry. □

Finally we state in property 2.1.5 some alternative names under which the operator norm can be found in the literature.

Property 2.1.5. *The operator norm is also known under the names of spectral norm, induced norm, or 2-norm.*

Matrix entry-wise norms and mixed norms

One of the major families of matrix norms is *element-wise* norms, that is, norms whose expressions are given by functions of the elements of a matrix. We will use element-wise norms extensively when working with the *structure* of a matrix rather than its spectral properties. We give the general expression of element-wise norms and mixed element-wise norms. The first type of entry-wise norms is ℓ_p norms, and can be defined as follows.

Definition 2.1.7. The entry-wise p norm of $\mathbf{A}$ is:

$$||\mathbf{A}||_p = ||\mathrm{vec}(\mathbf{A})||_p = \left(\sum_{i,j} |a_{ij}|^p\right)^{1/p}$$

As we can see from the definition, each element of the matrix is treated individually as would be the case for the ℓ_p norm on a vector. This means we can only influence *non-specific* structural properties of the matrix since there is no notion of neighbourhood or groups. Such a property can be desirable but can also limit our modelling power.

As a consequence, we present a more general form of element-wise norms called *mixed ℓ_p/ℓ_q norms*. These norms give us more flexibility; ℓ_p norms can be seen as special cases of ℓ_p/ℓ_q norms with $p = q$.

Definition 2.1.8. The entry-wise p, q norm of $\mathbf{A}$ is:

$$||\mathbf{A}||_{p,q} = \left(\sum_j \left(\sum_i |a_{ij}|^p\right)^{q/p}\right)^{1/q}$$

Matrix Schatten-p norms

The second major family of matrix norms is the family of *Schatten-p* norms, which contrary to element-wise norms are not defined as functions of the matrix elements but directly of the *singular values* of the matrix. Therefore, Schatten norms can be used to control spectral properties of matrices.

Definition 2.1.9. A Schatten-p norm is obtained by taking the p norm of the singular values:

$$||\mathbf{A}||_p = ||(\sigma_1, \ldots, \sigma_{\min(m,n)})||_p = \left(\sum_i^{\min(m,n)} \sigma_i^p \right)^{1/p}$$

An important property of Schatten norms is their invariance to unitary transformations, we say that the Schatten norms are *unitary invariant* and express it formally in Property 2.1.6.

Property 2.1.6. *Let $\mathbf{U} \in \mathcal{O}(n)$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$, then for any p, we have:*

$$||\mathbf{U}\mathbf{A}\mathbf{U}^\mathsf{T}||_p = ||\mathbf{A}||_p$$

Important special cases of matrix norms

We now present special cases of matrix norms of major importance in computational statistics and define a related object known as the ℓ_0 *pseudo*-norm.

ℓ_0 pseudo-norm The ℓ_0 pseudo-norm is not exactly a norm in the mathematical sense but is a measure of the density (and thus of the sparsity) of a matrix.

Definition 2.1.10. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, then $||\mathbf{A}||_0$ is the number of non-zero elements in $\mathbf{A}$.

ℓ_1 norm and the Lasso The Lasso is a widely used penalty for promoting sparse solutions and consists in putting ℓ_1 regularisation on the penalised variables. It induces sparsity in an unstructured way.

Definition 2.1.11. Let $\mathbf{A} \in \mathbb{R}^{n,r}$, $\mathbf{A} = (a_{ij})$, then $||\mathbf{A}||_1 = \sum_{i,j} |a_{ij}|$.

ℓ_1/ℓ_2 norm and the non-overlapping Group Lasso The ℓ_1/ℓ_2 norm is also known as the Group Lasso and induces structured sparsity by driving whole groups of variables to zero at once. We will assume that the groups do not overlap, even though group-sparsity inducing penalties have been studied for overlapping and hierarchical groups and efficient inference is possible [113, 59]. From the expression of mixed norms given in Definition 2.1.8 we see that on matrices, the matrix *columns* form the groups of variables.

Definition 2.1.12. Let $\mathbf{A} \in \mathbb{R}^{n,r}$, and $\mathbf{C} = (c_1, \ldots, c_N)$ a set of N groups such that every element of $\mathbf{A}$ belongs to exactly one c_i.

The ℓ_1/ℓ_2 norm is the ℓ_1 norm of the vector of the ℓ_2 norms of the groups: $||\mathbf{A}||_{\ell_1/\ell_2} = \sum_k ||c_k||_2$, hence the name of Group Lasso.

Another structured norm commonly used in place of the the ℓ_1/ℓ_2 norm is the ℓ_1/ℓ_∞ norm where the ℓ_∞ norm replaces the ℓ_2 norm. We refer to Mairal et al. [67] for more information on the subject.

Frobenius norm The Schatten-2 norm is most commonly called the Frobenius norm or Euclidean norm and is the canonical norm on matrix spaces.

Definition 2.1.13. Let $\mathbf{A} \in \mathbb{R}^{n,r}$, then $||\mathbf{A}||_{\mathrm{F}} = \sqrt{\mathrm{tr}\left[\mathbf{A}\mathbf{A}^{\mathsf{T}}\right]} = \sqrt{\mathrm{tr}\left[\mathbf{A}^{\mathsf{T}}\mathbf{A}\right]}$

Nuclear (trace) norm The Schatten-1 norm is called the nuclear norm.

Definition 2.1.14. Let $\mathbf{A} \in \mathbb{R}^{n,r}$, then $||\mathbf{A}||_*$ is the sum of the singular values of $\mathbf{A}$.

A key characterisation of the nuclear norm is given by theorem 2.1.15.

Theorem 2.1.15 (Variational characterisation of the nuclear norm)**.** *Let* $\mathbf{X} \in \mathbb{R}^{n\times m}$.
Suppose $\mathrm{rank}(\mathbf{X}) = r$. *Let* $\mathbf{U} \in \mathbb{R}^{n\times r}$ *and* $\mathbf{V} \in \mathbb{R}^{m\times r}$ *such that* $\mathbf{X} = \mathbf{U}\mathbf{V}^{\mathsf{T}}$*, then the following holds:*

$$||\mathbf{X}||_* = \underset{\mathbf{U},\mathbf{V}}{\mathrm{argmin}} \frac{1}{2}\left(||\mathbf{U}||_{\mathrm{F}}^2 + ||\mathbf{V}||_{\mathrm{F}}^2\right)$$

Proof. See Recht et al. [83]. □

Schatten-∞ norm The Schatten-∞ norm is the largest singular value of a matrix.

2.1.4 Vectorisation operator and Kronecker product

All the properties are given without proof, we refer to Schäcke [85] for more information.

Definition 2.1.16. The vectorisation operator vec transforms a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ in a vector $\mathrm{vec}(\mathbf{A}) \in \mathbb{R}^{nm}$ obtained by stacking the **columns** of $\mathbf{A}$:

$$\mathrm{vec}\left[\begin{pmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{pmatrix}\right] = (a_{11}\dots a_{n1}\dots a_{1m}\dots a_{nm})^{\mathsf{T}}$$

The Kronecker product defines a matrix product and is of great importance to the work developed so far.

Definition 2.1.17. Let $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{B} \in \mathbb{R}^{r\times p}$, the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$ is

$$\mathbf{K} = \mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & \dots & a_{1m}\mathbf{B} \\ \vdots & & \vdots \\ a_{n1}\mathbf{B} & \dots & a_{nm}\mathbf{B} \end{pmatrix}$$

We have $\mathbf{K} \in \mathbb{R}^{nr\times mp}$.

The Kronecker product has some interesting properties, some of them linked to the vec operator.

Property 2.1.7. *Let* $\mathbf{A}, \mathbf{B}, \mathbf{C}$ *three real matrices such that* $\mathbf{ABC}$ *exists. Then*

$$\mathrm{vec}(\mathbf{ABC}) = (\mathbf{C}^{\mathsf{T}} \otimes \mathbf{A})\mathrm{vec}(\mathbf{B})$$

Property 2.1.8. *The Kronecker product is compatible with usual matrix operations:*

$$(\mathbf{A} \otimes \mathbf{B})^\mathsf{T} = \mathbf{A}^\mathsf{T} \otimes \mathbf{B}^\mathsf{T}$$
$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$
$$(\mathbf{A} \otimes \mathbf{B})^\dagger = \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger$$

Where $\mathbf{A}^\dagger$ denotes the Moore-Penrose pseudoinverse of $\mathbf{A}$.

Property 2.1.9. *Trace and the Kronecker product:*

$$\mathrm{tr}\left[\mathbf{A} \otimes \mathbf{B}\right] = \mathrm{tr}\left[\mathbf{A}\right] \mathrm{tr}\left[\mathbf{B}\right]$$

Property 2.1.10. *The group of orthogonal matrices is stable by Kronecker product, i.e, if $\mathbf{A}$ and $\mathbf{B}$ orthogonal, then $\mathbf{A} \otimes \mathbf{B}$ is orthogonal.*

Property 2.1.11. *Let $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ such that $\mathbf{A}^\mathsf{T}\mathbf{B}\mathbf{C}\mathbf{D}^\mathsf{T}$ exists, then:*

$$\mathrm{tr}\left[\mathbf{A}^\mathsf{T}\mathbf{B}\mathbf{C}\mathbf{D}^\mathsf{T}\right] = \mathrm{vec}(\mathbf{A})^\mathsf{T}(\mathbf{C} \otimes \mathbf{B})\mathrm{vec}(\mathbf{D})$$

2.2 Multilinear algebra

In this thesis, and following the mathematical literature, the term *tensor* will refer to multidimensional - or N-dimensional - arrays. In the physics and engineering literature, a *tensor* is sometimes used as a short name for objects known in mathematics as *tensor fields*, which are unrelated to our work.

A tensor generalises the notion of vector and matrices to a higher number of dimensions. Following Kolda and Bader [55], we call the number of dimensions of a tensor its *order*, number of *modes*, or number of *ways*. As such, a scalar is a 0^{th} order tensor, a vector is a 1-way tensor, and a matrix a 2-way tensor.

Tensors of order greater or equal to 3 are generally referred to as *higher-order tensors*. The study of higher-order tensors and their properties constitutes the field of **multilinear algebra**.

An N-mode tensor of size I_1 along the first mode, I_2, along the second, up until I_N along the last mode resides in a *tensor space* K, which is the tensor product of N I_j-dimensional vector spaces. In this work, we will only consider spaces for which the associated scalar field is the field of real numbers $\mathbb{R}$. In addition, every n-dimensional vector space is isomorphic to $\mathbb{R}^n$, so we will simply write $K = \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$.

In this section, we describe the conventions used in this thesis when dealing with tensor objects, and state the basic mathematical properties of tensors required for our algorithms.

2.2.1 Notations

We follow the usual convention of the multilinear algebra literature and denote tensors by boldface calligraphic letters and the dimension of vector spaces in superscript. For instance, an N-mode tensor of dimensions $I_1, \ldots I_N$ along each mode would be $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$.

An element of $\boldsymbol{\mathcal{X}}$ is $x_{i_1,i_2,\ldots,i_N}$ with each i_j bounded between 1 and I_j. When referring to subsets of elements grouped by their indexes we will make use of notational short-cuts for brevity. $x_{i_1,\ldots,i_{j-1},:,i_{j+1},\ldots,i_N}$ refers to all the elements of $\boldsymbol{\mathcal{X}}$ with indexes $i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_N$ fixed along the corresponding modes, and the j^{th} index free to vary between 1 and I_j. Similarly we will use $i_j : k_j$ to denote the elements of $\boldsymbol{\mathcal{X}}$ with indexes between i and k on mode j.

2.2.2 Tensor basics

Several notions of linear algebra generalise nicely to tensors. We review them and give graphical illustrations for the reader's convenience. We also describe how tensors can be converted to matrices, and vectors as such operations are commonly used by algorithms for low-rank modelling on tensors.

Tensor fibres and slices

The mode-n fibres are all the subsets of $\boldsymbol{\mathcal{X}}$ obtained by fixing all but the n^{th} index to a given value. Fibres generalise the notion of matrix rows and columns. In fact, since a matrix *is* a 2-way tensor, columns and rows are the mode-1 and mode-2 fibres respectively. For 3-way tensors, following the literature (Kolda and Bader [55], Lu et al. [66]), we will refer to mode-3 fibres as *tensor tubes*. Figure 2.1 illustrates the concept more clearly for 3-way tensors.

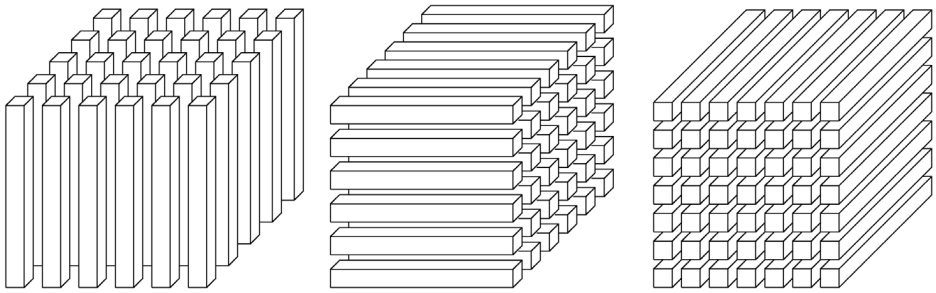

Figure 2.1: Illustration of mode-1, mode-2, and mode-3 fibres for a 3-way tensor with the usual convention for the ordering of the modes. Image adapted from Kolda and Bader [55].

A *tensor slice* is a 2-dimensional subset obtained by letting two indexes vary while fixing the remaining indexes to specific values. Tensor slices are best visualised for 3-way tensors, in which case they are obtained by "slicing" the tensor along one dimension. Figure 2.2 show the slices along each mode of a 3-way tensor.

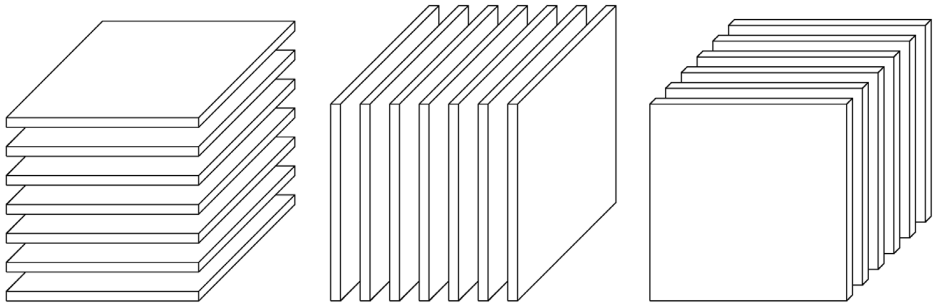

Figure 2.2: Illustration of mode-1, mode-2, and mode-3 slices for a 3-way tensor with the usual convention for the ordering of the modes. Image adapted from Kolda and Bader [55].

In the subsequent chapters of this thesis, we will build algorithms that work on the **frontal slices** of a 3-way tensor, which are the *mode-3* slices.

Tensor vectorisation and matricisation along a mode

An N-way tensor can be re-arranged in an M-way tensor with $1 \leq M \leq N$. The two most useful rearrangements are the tensor *vectorisation* and *matricisation*, which correspond to tensors of order

1 an 2 respectively. Since there are multiple ways of performing these re-arrangements, both in terms of the dimensions along the modes of the resulting tensor and of the possible permutations of the tensor elements, we give precise definitions of these operations as found in the literature.

Definition 2.2.1 (Tensor vectorisation)**.** Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an N^{th}-order tensor. The *tensor vectorisation operator* $\boldsymbol{\mathcal{X}} \mapsto \text{vec}(\boldsymbol{\mathcal{X}})$ maps each element $x_{i_1,i_2,\ldots,i_N}$ of $\boldsymbol{\mathcal{X}}$ to a unique element x_j of a real-valued vector $\text{vec}(\boldsymbol{\mathcal{X}})$ of dimension $\prod_{m=1}^{N} I_m$ with the following bijection:

$$j = 1 + \sum_{k=1}^{N} (i_k - 1)\, J_k \quad \text{and} \quad J_k = \prod_{m=1}^{k-1} I_m \tag{2.1}$$

Definition 2.2.2 (Matricisation along one mode)**.** Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an N^{th}-order tensor. The mode-n tensor *matricisation* - or *unfolding*, with $n \in \{1, 2, \ldots, N\}$, is the matrix $\mathbf{X}_{[n]}$ of dimensions $(I_n, \prod_{k \neq n} I_k)$ such that tensor entry $x_{i_1,i_2,\ldots,i_N}$ is mapped to a unique element $x_{i_n,j}$, with the following bijection:

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} (i_k - 1)\, J_k \quad \text{and} \quad J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \tag{2.2}$$

In other words, the mode-n matricisation stacks the mode-n fibres in a single matrix in column-major order by first varying I_1, then I_2, up until I_N with the exception of I_n.

Similarly, the tensor vectorisation is obtained by stacking the elements of $\boldsymbol{\mathcal{X}}$ in a single vector by first varying I_1, then I_2, up until I_N.

Definition 2.2.3 (mode-n tensorisation)**.** The inverse operator of the mode-n unfolding is the mode-n tensorisation and is denoted $\text{fold}_n(\mathbf{X}_{[n]}) = \boldsymbol{\mathcal{X}}$.

Intuitively, the mode-n tensorisation "folds back" the matricised tensor into its original tensor shape by applying the reciprocate of bijection 2.2.

The following example on a 3-way tensor should help the reader visualise the concepts of vectorisation and matricisation. Let $\boldsymbol{\mathcal{X}}$ be given by the following frontal slices in Matlab notation:

$$\boldsymbol{\mathcal{X}}(:,:,1) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad \boldsymbol{\mathcal{X}}(:,:,2) = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

Then the following hold true:

$$\text{vec}(\boldsymbol{\mathcal{X}}) = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)^{\mathsf{T}}$$

$$\mathbf{X}_{[1]} = \begin{pmatrix} 1 & 2 & 3 & 7 & 8 & 9 \\ 4 & 5 & 6 & 10 & 11 & 12 \end{pmatrix}$$

$$\mathbf{X}_{[2]} = \begin{pmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{pmatrix}$$

$$\mathbf{X}_{[3]} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 10 & 11 & 12 \end{pmatrix}$$

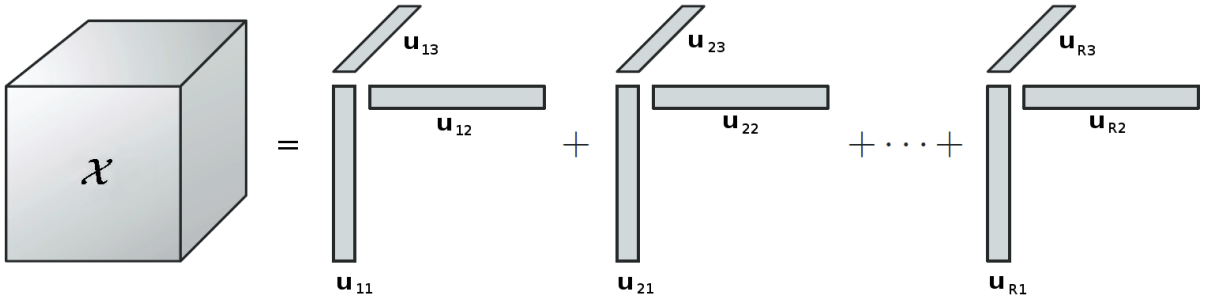

Figure 2.3: Illustration of the CP decomposition of the 3^{rd}-order tensor $\boldsymbol{\mathcal{X}}$ as the sum of rank-1 tensors $\mathbf{u}_{r1} \circ \mathbf{u}_{r2} \circ \mathbf{u}_{r3}$, for $r \in \{1, 2, \ldots, R\}$. Adapted from Kolda and Bader [55].

2.2.3 The CP and Tucker decompositions

We previously mentioned matrix decompositions - or factorisations - in section 2.1.2. In this section, we present two important *tensor* factorisations.

The PARAFAC/CANDECOMP (CP) decomposition

The CANDECOMP/PARAFAC (CP) decomposition was first proposed in 1927 by Hitchcock [49] and rediscovered independently in 1970 under the names CANDECOMP (CANonical DECOMPosition) [23] and PARAFAC (PARAllel FACtors) [47] in the psychometrics community. The CP decomposition can be seen as a generalisation of the matrix SVD and decomposes a tensor as the sum of *rank*-1 *tensors*.

Definition 2.2.4 (Rank-1 tensor)**.** A tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is said to be rank-1 if it can be written as the outer product of N vectors $\mathbf{u}_n \in \mathbb{R}^{I_n}$, for $i \in \{1, 2, \ldots, N\}$:

$$\boldsymbol{\mathcal{X}} = \mathbf{u}_1 \circ \mathbf{u}_2 \circ \ldots \circ \mathbf{u}_N \tag{2.3}$$

Equivalently, for every entry of $\boldsymbol{\mathcal{X}}$:

$$x_{i_1 i_2 \ldots i_N} = u_{i_1} u_{i_2} \ldots u_{i_N} \tag{2.4}$$

Definition 2.2.5 (CP decomposition)**.** Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an N^{th}-order tensor. The CP decomposition of $\boldsymbol{\mathcal{X}}$ is

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \mathbf{u}_{r1} \circ \mathbf{u}_{r2} \circ \ldots \circ \mathbf{u}_{rN} \tag{2.5}$$

With equality only for suitably large R, as we shall discuss in section 2.2.5.

Let $\mathbf{U}^{(n)} = \begin{bmatrix} \mathbf{u}_{1n} & \mathbf{u}_{2n} & \ldots & \mathbf{u}_{Rn} \end{bmatrix} \in \mathbb{R}^{I_n \times R}$ for $n \in \{1, 2, \ldots, N\}$ the n^{th} *factor matrix*, the CP decomposition is generally written as:

$$\boldsymbol{\mathcal{X}} \approx \mathbf{U}^{(1)} \circ \mathbf{U}^{(2)} \circ \ldots \circ \mathbf{U}^{(N)} \tag{2.6}$$

It can be convenient to choose $\mathbf{u}_{rn}$ of unit length $||\mathbf{u}_{rn}||_2 = 1$ and to write:

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} s_r \mathbf{u}_{r1} \circ \mathbf{u}_{r2} \circ \ldots \circ \mathbf{u}_{rN} \tag{2.7}$$

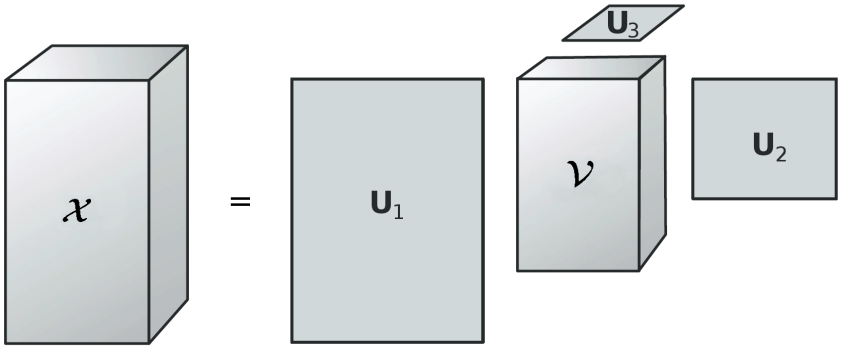

Figure 2.4: Illustration of the HOSVD on a 3^{rd}-order tensor $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{V}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3$. Image adapted from Kolda and Bader [55].

where s_r is a scalar that compensates for the magnitudes of vectors $\mathbf{u}_{rn}$. Fig. 2.3 illustrates the CP decomposition of a 3^{rd}-order tensor.

The Tucker/Higher Order SVD decomposition

The other major tensor decomposition is the Higher Order Singular Value Decomposition (HOSVD), also known as Tucker decomposition. It also dates back to Hitchcock [49] but owes its name to the 1966 work of Tucker [101] in the field of psychometrics where it was presented as a multidimensional extension of factor analysis. Kroonenberg [57] (1983) further developed the method, while De Lathauwer et al. [27] (2000) more recently presented it under the name HOSVD. In the same way the matrix SVD computes the left and right singular vectors, HOSVD recovers orthonormal bases for the spaces spanned by the mode-n fibres.

Definition 2.2.6 (HOSVD)**.** Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be an N^{th}-order tensor. The HOSVD of $\boldsymbol{\mathcal{X}}$ is

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{V}} \times_{i=1}^{N} \mathbf{U}^{(i)} \tag{2.8}$$

With:

- Every $\mathbf{U}^{(i)}$ is orthogonal
- Any two subtensors of the core tensor $\boldsymbol{\mathcal{S}}$ are orthogonal: $\langle \boldsymbol{\mathcal{S}}_{i_k=p}, \boldsymbol{\mathcal{S}}_{i_k=q} \rangle = 0$ for $p \neq q$. $\boldsymbol{\mathcal{S}}_{i_k=p}$ is obtained by fixing the indices of mode k to p.
- The subtensors of $\boldsymbol{\mathcal{S}}$ are ordered according to their Frobenius norm: $\forall k, ||\boldsymbol{\mathcal{S}}_{i_k=1}||_{\text{F}} \geq \ldots \geq ||\boldsymbol{\mathcal{S}}_{i_k=n_k}||_{\text{F}} \geq 0$.

In the above definition, $\boldsymbol{\mathcal{V}}$ is called the *core tensor*. Fig. 2.4 illustrates the HOSVD of a 3^{rd}-order tensor.

If the core tensor is diagonal, then the HOSVD reduces to the CP decomposition with orthogonal factor matrices. For 3-way tensors, the *Tucker2* decomposition is a simple restriction of the *Tucker* decomposition where $\mathbf{U}^{(3)} = \mathbf{I}$.

2.2.4 Tensor norms and SVDs

In this section we present extensions of the matrix norms to tensors, as well as different definitions of the tensor SVD that are found in the literature.

Tensor inner product and the tensor Frobenius norm

The notion of vector dot product generalises naturally to tensors of compatible sizes. We define the tensor dot product (or inner product) by:

Definition 2.2.7 (Tensor inner product)**.** Let $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, then

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle = \sum_{i_1, \ldots, i_N} x_{i_1 \ldots i_N} y_{i_1 \ldots i_N}$$

All the usual properties of an inner product hold. We can now define the tensor Frobenius norm as the canonical norm associated with the tensor inner product.

Definition 2.2.8 (Tensor Frobenius norm)**.** Let $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$, then

$$||\boldsymbol{\mathcal{X}}||_{\mathrm{F}} = \sqrt{\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle} = \sqrt{\sum_{i_1, \ldots, i_N} x_{i_1 \ldots i_N}^2}$$

Tensor ℓ_p norm and mixed norms

The tensor ℓ_p norm is defined as a direct extension of the matrix ℓ_p norm:

$$||\boldsymbol{\mathcal{X}}||_p = ||\mathrm{vec}(\boldsymbol{\mathcal{X}})||_p$$

Similarly to matrix mixed (or structured) norms, we can define tensor mixed norms. We give the example of the tensor $\ell_1/\ell_1/\ell_2$ norm for 3-way tensors as found in [115]:

$$||\boldsymbol{\mathcal{X}}||_{1,1,2} = \sum_{i,j} ||\boldsymbol{\mathcal{X}}(i, j, :)||_{\mathrm{F}}$$

Tensor spectral norm and tensor SVD

Building on the notion of rank-one tensor, we define the tensor spectral norm as found in Derksen [28]. This notion of spectral norm is a direct extension of the matrix spectral norm - or operator norm - except that unit-norm rank-one tensors replace the vectors found in the definition of the matrix spectral norm. The tensor spectral norm is then defined as the maximum value of $|\langle \boldsymbol{\mathcal{X}}, u \rangle|$ with u a normal rank-one tensor:

Definition 2.2.9 (Tensor spectral norm)**.** The tensor spectral norm is defined as:

$$||\boldsymbol{\mathcal{X}}||_2 = \max_{\forall i,\, ||y_i||=1} \langle \boldsymbol{\mathcal{X}}, y_1 \otimes \ldots \otimes y_N \rangle$$

The definition of SVD consistent with this definition of spectral norm is the Higher Order SVD.

Defining the tensor t-SVD

Recent work in computer vision such as Lu et al. [66], Zhang et al. [115], Hu et al. [50] define tensor analogues to the matrix product, matrix SVD, and hence tensor nuclear norm as will be discussed in section 2.2.5. These notions have been developed for 3-way tensors as they are the most common tensors in image and video processing. The key underlying idea is that a 3-way tensor can be seen as a matrix of tensor *tubes*, *i.e,* of mode-3 fibres. This makes more sense if we consider the special case of an RGB image: each pixel of the image will have three components, so

we can represent a pixel as a vector of $\mathbb{R}^3$. The image is then a 3-way tensor with each frontal slice representing one of the three colour channels. Such an interpretation can of course be generalised to other kind of data where a third dimension is natural: the different values over time of one pixel in a video, the multiple spectra of a multispectral image, etc. These new tools operate on the frontal slices of tensors and are often best expressed in the Fourier domain. We refer the reader to one of the many available references on Fourier analysis for a better understanding of the *Discrete Fourier Transform*, discrete and circular convolutions, and how they are related.

We first introduce some block-based operators that act on the frontal slices of a tensor and are required for the definition of the tensor t-product and of the t-SVD.

Definition 2.2.10 (bcirc operator)**.** The block-circulant matrix $\mathrm{bcirc}(\boldsymbol{\mathcal{X}})$ is obtained from $\boldsymbol{\mathcal{X}}$ by construction:

$$\mathrm{bcirc}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \mathbf{X}^{(1)} & \mathbf{X}^{(I_3)} & \dots & \mathbf{X}^{(2)} \\ \mathbf{X}^{(2)} & \mathbf{X}^{(1)} & \dots & \mathbf{X}^{(3)} \\ \vdots & & & \vdots \\ \mathbf{X}^{(I_3)} & \mathbf{X}^{(I_3-1)} & \dots & \mathbf{X}^{(1)} \end{bmatrix}$$

With I_3 the dimension on the third mode.

The motivation for relying on block-circulant matrices is that they can be efficiently diagonalised by the discrete Fourier transform.

Definition 2.2.11 (Block-vectorisation of the frontal slices)
The block vectorisation of the frontal slices and its inverse operator are defined by:

$$\mathrm{bvec}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \mathbf{X}^{(1)} \\ \mathbf{X}^{(2)} \\ \vdots \\ \mathbf{X}^{(I_3)} \end{bmatrix}$$

And $\mathrm{bvfold}(\mathrm{bvec}(\boldsymbol{\mathcal{X}})) = \boldsymbol{\mathcal{X}}$.

Definition 2.2.12 (Block diagonal matrix of the frontal slices)
The block-diagonal matrix $\mathrm{bdiag}(\boldsymbol{\mathcal{X}})$ *is obtained by construction from the frontal slices of* $\boldsymbol{\mathcal{X}}$*:*

$$\mathrm{bdiag}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} \mathbf{X}^{(1)} & & \\ & \ddots & \\ & & \mathbf{X}^{(I_3)} \end{bmatrix}$$

And $\mathrm{bdfold}(\mathrm{bdiag}(\boldsymbol{\mathcal{X}})) = \boldsymbol{\mathcal{X}}$.

Based on these definitions we define tensor analogues for the matrix product and the matrix Singular Value Decomposition.

Definition 2.2.13 (Tensor t-product)**.** The *t-product* of two tensors $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_2 \times I_4 \times I_3}$ is:

$$\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}} = \mathrm{bvfold}(\mathrm{bcirc}(\boldsymbol{\mathcal{X}})\mathrm{bvec}(\boldsymbol{\mathcal{Y}}))$$

$\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_4 \times I_3}$.

The mode-3 fibres of $\boldsymbol{\mathcal{M}}$ are obtained by circular convolutions of the mode-3 fibres of $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$:

$$\boldsymbol{\mathcal{M}}(i,j,:) = \sum_{k=1}^{I_2} \boldsymbol{\mathcal{X}}(i,k,:) * \boldsymbol{\mathcal{Y}}(k,j,:).$$

An the t-product corresponds to the matrix multiplication of the frontal slices in the Fourier domain. Note here that $*$ denotes the convolution operation and not the Hadamard product, for which we will use the $\circledast$ convention.

Before defining the tensor t-SVD, it is necessary to extend the notions of orthogonal matrices and matrix transpose to tensors, which can be done as follows.

Definition 2.2.14 (Tensor transpose)**.** Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the transpose tensor $\boldsymbol{\mathcal{X}}^\mathsf{T}$ is obtained by transposing each frontal slice $\mathbf{X}^{(i)}$ and reversing the order of the transposed frontal slices from 2 to I_3.

Note that the previous definition is straightforwardly extended to complex-valued tensors by considering the *conjugate transpose* instead of the transpose.

Definition 2.2.15 (Identity tensor)**.** The identity tensor $\boldsymbol{\mathcal{I}}$ is a tensor whose first frontal slice is the identity and all other frontal slices are zero.

Definition 2.2.16 (Orthogonal tensor)**.** A real-valued tensor $\boldsymbol{\mathcal{X}}$ is orthogonal iff

$$\boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{X}}^\mathsf{T} = \boldsymbol{\mathcal{X}}^\mathsf{T} * \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{I}}$$

Finally we obtain an extension of the SVD theorem to tensors and the tensor t-SVD which characterises the optimal approximating tensors of a certain form.

Definition 2.2.17 (Tensor t-SVD)**.** The tensor SVD (t-SVD) of $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined by:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{S}} * \boldsymbol{\mathcal{V}}^\mathsf{T}$$

With $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{I_1 \times I_1 \times I_3}$ and $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{I_2 \times I_2 \times I_3}$ orthogonal.

Theorem 2.2.18. *Let the t-SVD of $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ be given by $\boldsymbol{\mathcal{M}} = \boldsymbol{\mathcal{U}} * \boldsymbol{\mathcal{S}} * \boldsymbol{\mathcal{V}}^\mathsf{T}$ and for $k < \min(I_1, I_2)$ define $\boldsymbol{\mathcal{M}}_k = \sum_{i=1}^{k} \boldsymbol{\mathcal{U}}(:,i,:) * \boldsymbol{\mathcal{S}}(i,i,:) * \boldsymbol{\mathcal{V}}^\mathsf{T}(:,i,:)$, then*

$$\boldsymbol{\mathcal{M}}_k = \arg\min_{\tilde{\boldsymbol{\mathcal{M}}} \in \mathbb{M}} ||\boldsymbol{\mathcal{M}} - \tilde{\boldsymbol{\mathcal{M}}}||_\mathrm{F}$$

Where $\mathbb{M} = \{\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{Y}} | \boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times k \times I_3}, \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{k \times I_2 \times I_3}$.

2.2.5 Tensor ranks and their convex surrogates

Unlike the matrix rank, defining the rank of a tensor is a difficult problem and several different definitions exist. In this section, we give an overview of the most commonly found definitions in the literature.

The first definition of the tensor rank that appeared is the *tensor CP rank*, also known as *Kruskal rank*, or often simply the tensor rank. We will refer to it by $\mathrm{rank}_\mathrm{CP}$ and give its definition below.

Definition 2.2.19 (Tensor CP-rank)**.** The tensor CP rank R_CP is the smallest R such that equation 2.5 from definition 2.2.5 is **exact**.

That is, the smallest number of rank-one tensors needed to represent the tensor exactly. Computing the CP-rank of a tensor is an NP-hard problem, and methods that directly aim at minimising the CP-rank are rare.

A much more tractable definition of the tensor rank is the *Tucker rank*. Let us first define the tensor n-rank as the rank of the mode-n matricisation, then the Tucker rank is defined as the **vector** of the n-ranks for all n.

Definition 2.2.20 (Tensor Tucker rank)**.** Let $\boldsymbol{\mathcal{X}}$ an N-way tensor, then the tensor Tucker rank of $\boldsymbol{\mathcal{X}}$, is the vector

$$\text{rank}_\text{T}(\boldsymbol{\mathcal{X}}) = (\text{rank}(\mathbf{X}_{[1]}), \ldots, \text{rank}(\mathbf{X}_{[N]}))$$

Where $\text{rank}()$ is the matrix rank.

The name Tucker rank comes from the following property, which gives the relationship between the Tucker factors in the HOSVD, and the rank of the tensor matricisations.

Property 2.2.1. *Assuming the following Tucker decomposition for a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$:*

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{V}} \times_{i=1}^{N} \mathbf{U}^{(i)}$$

Then

$$\forall 1 \leq i \leq N,\ \text{rank}(\mathbf{X}_{[i]}) = \text{rank}(\mathbf{U}^{(i)})$$

Lu et al. [66], Zhang et al. [115], Hu et al. [50] rely on the notion of *Tensor Tubal Rank*. Let us first introduce the tensor $\hat{\boldsymbol{\mathcal{X}}}$ obtained from $\boldsymbol{\mathcal{X}}$ by taking the Discrete Fourier Transform along the third mode; we define the *tensor tubal rank* as well as the *tensor multi rank*.

Definition 2.2.21 (Tensor multi rank)**.** The tensor multi-rank of $\boldsymbol{\mathcal{X}}$ is the vector of the ranks of the frontal slices of $\hat{\boldsymbol{\mathcal{X}}}$.

Definition 2.2.22 (Tensor tubal rank)**.** The tensor tubal rank defined as the number of non-zero singular tubes in the tensor t-SVD.

Towards a Tensor Nuclear Norm

Inspired by the use of the matrix nuclear norm as the tightest convex surrogate of the matrix rank, several approaches have been proposed in the literature to define a *Tensor nuclear norm*. In this section, we summarise the existing approaches and discuss which notion of tensor rank they approximate.

The first approach to defining a tensor nuclear norm is based on the nuclear norm of the tensor mode-n matricisations and provides a convex surrogate to the Tucker rank. The typical choices are to define the tensor nuclear norm as either the sum of the nuclear norms of the unfoldings, or as a convex combination of them:

$$||\boldsymbol{\mathcal{X}}||_* = \sum_n ||\mathbf{X}_{[n]}||_*$$

This approach has been well studied as we will show in section 3.2.1 (references provided in this section), but is not a tight convex relaxation of the Tucker rank (Romera-Paredes and Pontil [84]).

Another definition of the nuclear norm is often found in the applied mathematics literature and is a direct analogue of the matrix nuclear norm. This definition of the tensor nuclear norm comes with a strongly related definition of the tensor spectral norm; in fact, these two norms are **dual**. We shall refer the reader to one of the many available resources on duality; such as *Convex Optimization* by Boyd and Vandenberghe [17].

Definition 2.2.23 (Tensor nuclear norm)**.** The tensor nuclear norm is defined as the dual of the tensor spectral norm and has closed-form expression:

$$||\boldsymbol{\mathcal{X}}||_* = \inf\{\sum_{p=1}^{+\infty} \sigma_p | \boldsymbol{\mathcal{X}} = \sum_{p=1}^{+\infty} \sigma_p \boldsymbol{\mathcal{X}}_p, \sigma_p \geq 0\}$$

Where $\boldsymbol{\mathcal{X}}_p$ are rank-one tensors of unit Frobenius norm.

This definition of nuclear norm is a suitable surrogate for the tensor CP-rank. However, computing it is NP-hard and difficult to approximate, so the methods that rely on this definition of the nuclear norm usually optimise it indirectly by working with the tensor spectral norm. We will present some of these dual approaches in section 3.2.4.

More recently, a definition of the tensor nuclear norm has been proposed based on the frontal slices in the Fourier domain. Zhang et al. [115] gives the following definition.

Definition 2.2.24 (Tensor Nuclear Norm)**.** Let $\boldsymbol{\mathcal{X}}$ a tensor and $\hat{\boldsymbol{\mathcal{X}}}$ the tensor obtained by taking the Discrete Fourier Transform (DFT) along the third mode of $\boldsymbol{\mathcal{X}}$, denoting by $\hat{\mathbf{X}}^{(k)}$ the k^{th} frontal slice of $\hat{\boldsymbol{\mathcal{X}}}$, then the Tensor Nuclear Norm is defined as:

$$\text{TNN}(\boldsymbol{\mathcal{X}}) = \sum_k ||\hat{\mathbf{X}}^{(k)}||_*$$

Which is the tightest convex relaxation of the multi-rank. Lu et al. [66] give a slightly different definition in that they define the TNN as the average of the nuclear norms of the frontal slices in the Fourier domain, in this case the notion of rank approximated is that of the tensor *average rank, i.e,* the average value of the tensor multi-rank.

2.3 Convex optimisation

In this section we assume basic familiarity with convex analysis and invite the reader to consult one of the many high-quality references on the subject when necessary such as Boyd and Vandenberghe [17], Bubeck [18]. In particular, we shall not redefine the notion of convex set, convex function, or convex envelop (also known as convex hull).

2.3.1 Proximal operators

Proximal operator are powerful tools in non-smooth optimisation and generalise the notion of projection. We first define the Moreau envelope of a function, the Moreau proximity operator, and its scaled version. We then give a geometric interpretation of the proximal mapping to help the reader get an intuition of the concept.

Definition 2.3.1. The *Moreau envelope* of f is given by:

$$g(y) = \min_x f(x) + \frac{1}{2}||x - y||_2^2 \tag{2.9}$$

So the Moreau envelope of a function f at a point y is obtained by minimising the sum of f and of a quadratic penalty between y and all possible other points x in the domain. The proximal operator, or proximal mapping, is defined as the arg-minimum of equation (2.9); a handy definition that arises often in practice is that of the *scaled* proximal operator, which is a straightforward extension to include a scaling factor.

Definition 2.3.2. The *Moreau proximity operator* of f is given by:

$$\text{prox}_f(y) = \underset{x}{\text{argmin}}\, f(x) + \frac{1}{2}||x - y||_2^2$$

Definition 2.3.3. The *Scaled Moreau proximity operator* of f is given by:

$$\text{prox}_{\lambda f}(y) = \underset{x}{\text{argmin}}\, \lambda f(x) + \frac{1}{2}||x - y||_2^2$$

Where the Moreau envelope is defined as the minimal *value* of $f(x) + \frac{1}{2}||x - y||_2^2$ with respect to x, the proximal operator returns the optimal x *itself*.

Geometrically speaking, we find the point x which minimises f while being close to y, as enforced by the squared distance between x and y. Therefore, the proximal operator *drives* points to the minimum of f. It should also be clear that any minimum of f is a stationary point of the proximal operator.

2.3.2 Augmented Lagrangian Function

We first define the Lagrangian function for an equality-constrained optimisation problem (P):

$$\begin{aligned} (P) \quad \min \quad & f(x) \\ \text{s.t} \quad & g(x) = 0 \end{aligned}$$

Definition 2.3.4. The Lagrangian of (P) is defined by:

$$L(x, \lambda) = f(x) - \langle \lambda, g(x) \rangle$$

Formulating the Lagrangian of (P) allows us to transform a constrained problem into an equivalent unconstrained problem:

$$(P') \quad \min \quad L(x, \lambda)$$

We have the following property that gives the first order optimality condition (necessary condition), knowing that since (P') is unconstrained, every feasible point is interior.

Property 2.3.1. *(First order necessary condition for the Lagrangian). Every solution x^* of (P) is a critical point of L, i.e,*

$$\nabla L(x^*, \lambda^*) = 0$$

Solving a problem using the standard Lagrangian function requires that we find an optimal value of the Lagrange multiplier λ, which can be difficult. In this case, we can instead devise algorithms that solve the **Augmented Lagrangian**.

Definition 2.3.5 (Augmented Lagrangian (AL))**.** The Augmented Lagrangian of (P) is defined by:

$$L(x, \lambda) = f(x) - \langle \lambda, g(x) \rangle - \frac{1}{2}\mu||g(x)||_2$$

We note that the augmented Lagrangian differs from the Lagrangian by the addition of a quadratic penalty with weight μ that penalises the violation of the constraint g.

The Augmented Lagrangian is a convexification of the Lagrangian. Methods based on the AL build a sequence λ^t such that $\lambda^{t+1} = \lambda^t - \mu^t g(x^{t+1})$. If the method converges, then under the right conditions the critical points of the Augmented Lagrangian are the critical points of the Lagrangian. Convergence is guaranteed for convex problems but the method may not converge in general.

2.3.3 Method of Alternating Direction of Multipliers and variable splitting

Here we give a succinct description of the ADMM method without proving the convergence of the generic algorithm.

Let (P) be an optimisation problem of the form:

$$\begin{aligned}(P) \quad \min \quad & f(x) + g(x) \\ \text{s.t} \quad & h(x) = 0\end{aligned}$$

The ADMM method minimises the Augmented Lagrangian by alternating minimisation over each variable (*i.e,* each direction).

In its simplest for, variable (or operator) splitting consists in replacing (P) by an equivalent problem (P') obtained by introducing an auxiliary variable z and the constraint $x = z$ so that the problem becomes separable:

$$\begin{aligned}(P) \quad \min \quad & f(x) + g(z) \\ \text{s.t} \quad & h(x) = 0 \\ & x = z\end{aligned}$$

x and z are optimised alternatively and the addition of the constraint ensures that they converge to the same value.

2.4 Multivariate statistics and Bayesian inference

The previous sections set the bases for tensor manipulations and convex optimisation-based tensor algorithms. We now review some prerequisites for a Bayesian treatment of tensor factorisations. We first present important matrix-variate distributions, than we define the Kullback-Leibler divergence as a measure of discrepancy between probability distributions, and briefly introduce approximate Bayesian inference in graphical model with the mean-field approximation.

2.4.1 Some matrix-valued probability distributions

We briefly define two matrix-valued distribution of interest to the work presented in this thesis. The first is the *Matrix Normal Distribution*, which generalises the Multivariate Normal to the matrix setting:

Definition 2.4.1. A random variable $\mathbf{X}$ is said to have a **Matrix Normal distribution** with mean $\mathbf{M}$, column covariance $\mathbf{\Sigma}_C$ and row covariance $\mathbf{\Sigma}_R$ if $\mathrm{vec}(\mathbf{X})$ has distribution $\mathcal{N}(\mathrm{vec}(\mathbf{M}), \mathbf{\Sigma}_R \otimes \mathbf{\Sigma}_C)$.

Property 2.4.1. *If $\mathbf{X} \in \mathbb{R}^{n \times m}$, the density of the matrix-variate Normal is [53]:*

$$f_{\mathbf{X}}(\mathbf{Q}) = \frac{1}{(2\pi|\mathbf{\Sigma}_C|^n|\mathbf{\Sigma}_R|^m)^{\frac{1}{2}}} \exp\left(\mathrm{tr}\left[-\frac{1}{2}\mathbf{\Sigma}_C^{-1}(\mathbf{Q}-\mathbf{M})\mathbf{\Sigma}_R^{-1}(\mathbf{Q}-\mathbf{M})^\mathsf{T}\right]\right)$$

We refer to Gupta and Nagar [45] for an extensive discussion on the properties of the Matrix Normal distribution.

The Wishart distribution is the matrix equivalent to the scalar Gamma distribution (and therefore of the χ^2 distribution for integer degrees of freedom). Since its introduction by Wishart in 1928, the Wishart distribution has been the subject of theoretical studies and has been extended to the non-central case.

We state two core properties and refer the reader to Anderson [5], Gupta and Nagar [45] for further discussion.

Property 2.4.2. *If* $\mathbf{X} \sim \mathcal{N}_{p,n}(\mathbf{0}, \mathbf{I}, \mathbf{I})$ *then* $\mathbf{X}\mathbf{X}^\mathsf{T} \sim \mathcal{W}(n, \mathbf{I})$

Property 2.4.3. *If* $\mathrm{vec}(\mathbf{X}) \sim \mathcal{N}_{p,n}(\mathbf{0}, \mathbf{\Sigma} \otimes \mathbf{I})$ *then* $\mathbf{X}\mathbf{X}^\mathsf{T} \sim \mathcal{W}(n, \mathbf{\Sigma})$

Property 2.4.4 ((Non-central Wishart distribution)**.** *If* $\mathrm{vec}(X) \sim \mathcal{N}_{p,n}(\mathbf{M}, \mathbf{\Sigma} \otimes \mathbf{I})$ *then* $\mathbf{X}\mathbf{X}^\mathsf{T} \sim \mathcal{W}(n, \mathbf{\Sigma}, \Theta)$
With $\Theta = \mathbf{\Sigma}^{-1}\mathbf{M}\mathbf{M}^\mathsf{T}$

The Wishart and non-central Wishart distributions are therefore the distributions of mean-centered and non-centered covariance matrices. Properties of the trace of a non-central Wishart random matrix can be found in Mathai and Pillai [70], Kourouklis and Moschopoulos [56].

We shall now proceed to presenting some concepts of approximate Bayesian inference in graphical models. We adopt the conventions used in Fox and Roberts [37] and Bishop [15, chap. 10] and will only present the case of unstructured mean-field approximation.

2.4.2 Kullback-Leibler divergence

The Kullback-Leibler (KL) divergence is not a *distance* in the mathematical sense, but defines a metric on probability distributions and as such gives a measure of how - intuitively - *close* two distributions are. A mathematically rigorous treatment of the geometry of probability distributions can be found in Amari and Nagaoka [3].

Definition 2.4.2. Let P and Q two probability distributions. The KL divergence of P and Q is defined as:

$$\mathrm{KL}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

Property 2.4.5. *The following two properties hold:*

$$\begin{aligned} &\mathrm{KL}(P||Q) \geq 0 \\ &\mathrm{KL}(P||Q) = 0 \quad \text{iff} \quad P = Q \end{aligned}$$

Property 2.4.6. *The KL divergence is* ***not*** *symmetric, so* ***generally****:*

$$\mathrm{KL}(P||Q) \neq \mathrm{KL}(Q||P)$$

2.4.3 Mean-field approximation in directed graphical models

Prior to introducing the mean-field approximation, we define the Markov blanket of a node in a graphical model Bishop [15], Murphy [73].

Definition 2.4.3. The Markov blanket of a node N is the smallest set $\mathcal{MB}(N) \subseteq \mathcal{V}$ of nodes such that

$$N \perp (\mathcal{V} \setminus \mathcal{MB}(N)) \mid \mathcal{MB}(N)$$

i.e, N is conditionally independent to every other node when the nodes in its Markov blanket are instantiated.

Note that the above definition doesn't assume orientation in the graphical model. Since we will be interested in inference in *directed* graphical models (DGM), we present characterisation of the Markov blanket of a node in a DGM in Property 2.4.7:

Property 2.4.7. *In a directed graphical model, the Markov blanket of a node is made of:*

- *The node's parents*
- *The node's children*
- *The other parents of the node's children (the co-parents)*

We refer to [15, 73] for a detailed discussion.

We can now present the mean-field approximation based on the notion of Markov blanket. Let $\mathbf{x}$ be a vector of random variables. Suppose the true joint distribution $P(\mathbf{x})$ unknown. Variational Inference aims at finding a tractable distribution Q that approximates P by optimising over a restricted space of possible distributions. **Variational Inference is a deterministic approximation.**

Various forms of variational inference exist, such as constraining the approximating distribution to a given family of parametric distributions, or - and this is the approach we take - assuming that Q factorises over the $\mathbf{x}_i$-s.

Definition 2.4.4. In mean-field Variational Inference, we assume the following form for Q:

$$Q(x) = \prod_i Q_i(\mathbf{x}_i)$$

Which we often write

$$Q(x) = \prod_i Q(\mathbf{x}_i)$$

We state the following property and refer the reader to Bishop [15], Fox and Roberts [37] for the proof.

Proposition 2.4.1. *The optimal approximating distribution for $\mathbf{x}_i$ satisfies:*

$$\log Q(\mathbf{x}_i)^* = \langle \log P(\mathbf{x}) \rangle_{\prod_{y \in \mathcal{MB}(\mathbf{x}_i)} Q(y)}$$

Finally, Variational Inference is tightly linked to the EM algorithm [1] (*c.f* Variational Free Energy and EM, Bishop [15]) and gives us an iterative algorithm where we update each variable in turn in a similar way to Gibbs sampling.

2.5 Image quality measures

In this section we introduce the measures we use to assess the quality of reconstructed images. Starting from the simplest cases we describe the motivations for more complex and realist metrics. We refer the reader to Mohammadi [72] for a recent (2014) survey of image quality measures.

2.5.1 Measures of pixel-wise similarity

Relative ℓ_2 error

Given two tensors $\boldsymbol{\mathcal{O}}$ (original) and $\boldsymbol{\mathcal{R}}$ (reconstructed), we define the relative error measured in terms of the ℓ_2 norm by:

$$\frac{||\boldsymbol{\mathcal{O}} - \boldsymbol{\mathcal{R}}||}{||\boldsymbol{\mathcal{O}}||}$$

Where the norm is the Frobenius norm defined as the square root of the tensor inner product $\sqrt{\langle \boldsymbol{\mathcal{R}}, \boldsymbol{\mathcal{R}} \rangle}$.

The relative error is a very simple measure and gives an idea of how close two tensors are *topologically*. It is not - however - designed to assess how *visually* close two images are.

Mean Squared Error and Peak Signal to Noise Ratio (PSNR)

A related measure to the relative error is the Mean Squared Error, which is the mean of the squared differences between each elements of two tensors. To put it differently, let $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ two identically-sized tensors of N elements in total, then

$$\mathrm{MSE}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}) = \frac{1}{N}||\mathrm{vec}(\boldsymbol{\mathcal{X}}) - \mathrm{vec}(\boldsymbol{\mathcal{Y}})||_2^2$$

The MSE represents the energy of the error signal. The PSNR is given in Decibels and depends on the dynamic range of the image. We give below the simplest definition in the case of monochromatic images

$$\mathrm{PSNR} = 10 \log \left(\frac{I_{Max}^2}{\mathrm{MSE}} \right)$$

Where I_{Max} is the maximum value a pixel can take.

2.5.2 Measures of structural similarity

MSE and PSNR do not to correlate well with subjective quality ratings, as can be seen in Figure (2.5). This motivates the introduction of more complex measures based on assumptions about the human visual system.

Figure 2.5: An example of when PSNR is not a good indication of the subjective quality of an image. PSNR of image labeled "Entry 2" is 22.58, while PSNR of "Entry 17" is 22.56. We would think the quality would be equivalent, but all the details have been removed from "Entry 17". Figure generated by experiment.

Structured Similarity Index (SSIM)

The idea of SSIM (Wang et al. [105]) stems from the assumption that the human eye is more sensitive to changes in the structure of an image rather than pixel-wise changes, and that nearby pixels are correlated. The index combines sub-measurements taken from multiple windows of identical size $N \times N$ as given by the following formula for two windows x and y:

$$\mathrm{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.10}$$

μ_x, σ_x^2 and σ_{xy} are the empirical mean, variance and covariance between the two windows. c_1 and c_2 are constants that depend on the dynamic range of the image. SSIM is therefore a quality measure designed for gray-scale images. The default choice for the window is Gaussian.

SSIM on its own is a *single scale* measure, and its value depends on the scale taken. The author provides[1] a rule of thumb for typical viewing distances, and a multi-scale extension of SSIM - called MS-SSIM - has been proposed (Wang and Bovik [104]) to alleviate this problem.

Some authors (Dosselmann and Dong Yang [32]) have challenged the performance of SSIM and derived its connection with the - often unreliable - MSE. Therefore, we will provide SSIM as an indicative metric of local structural similarity on the full-size, non-downsampled images, mainly as a reference index due to its popularity.

Feature Similarity Index (FSIM)

The Feature Similarity Index has been proposed in 2011 by Lin Zhang et al. [64] and is designed to reproduce more closely the perception of the human visual system (HVS). As the measure is quite involved and sophisticated, we refer to the original paper and to Mohammadi [72] for detailed explanations.

FSIM is also a metric designed for gray-scale image; however, the authors have proposed FSIMc for quality assessment on colour images.

2.5.3 Spectral Angle Mapper

The SAM has its roots in satellite imaging and remote sensing.

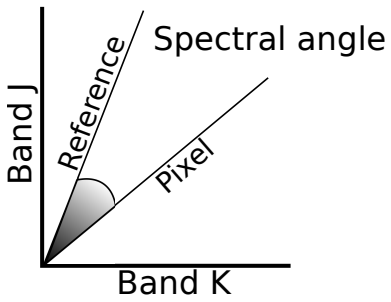

Figure 2.6: Representation of Reference Angle. Adapted from Rashmi et al. [81].

The algorithm computes angles between the spectral directions of two images. The lower the angles, the closer the images. We give for completeness the equation used to compute the spectral angle between two N-band images and refer to Rashmi et al. [81] (2014) for further information. Let us denote by r the spectrum of the reference image, and by t the spectrum of the compared image, then the spectral angle α is given by:

$$\alpha = \arccos\left(\frac{\langle r,\, t\rangle}{\sqrt{\langle r,\, r\rangle}\sqrt{\langle t,\, t\rangle}}\right) = \arccos\left(\frac{\langle r,\, t\rangle}{||r||_2||t||_2}\right)$$

[1] See http://www.cns.nyu.edu/ lcv/ssim/

Chapter 3

Low-rank modelling on matrices and tensors: a literature review

3.1 Low-rank modelling on matrices

In all subsequent subsections, $\mathcal{X}$ will denote a set of N observations $\mathbf{x}_n \in \mathbb{R}^p$, and $\mathbf{X}$ will be the matrix of **column** observations $\mathbf{X} = [\mathbf{x}_1 | \dots | \mathbf{x}_N]$ ($\mathbf{X} \in \mathbb{R}^{p \times N}$). We describe two broad families of algorithms: optimisation-based and Bayesian approaches, and identify the methods either by their generally-accepted names or by the title of the articles that introduced them.

3.1.1 Optimisation-Based Matrix decompositions

Low-rank modelling translates naturally as an optimisation problem: we seek the minimum rank solution that satisfies problem-dependent constraints. As a consequence, optimisation-based methods are the historical approach to rank-minimisation and are among the most successful and widely used methods. They have the advantage of giving the practitioner control on the algorithm's behaviour, and are generally capable of high accuracy. The downside to this flexibility is their sensitivity to the choice of hyper-parameters, and most of them need tuning to achieve the best performance.

In this subsection, we present two key optimisation-based matrix methods. The first, *Principal Component Analysis*, can be considered the origin of low-rank modelling; the second, *Robust Principal Component Analysis*, has been a major breakthrough in the field of *compressed sensing*. We also present work done on robust 2D methods that bears resemblance with the work we will develop in Chapter 4.

Principal Component Analysis (PCA)

PCA [78] in its simplest form aims at finding a **projection** on a linear embedding entirely determined by an **orthonormal** basis of vectors called the *Principal Components*. The goal behind PCA is to reduce the dimensionality of the data, *i.e,* to find *low-dimensional* linear embeddings. As not all projections are equivalent, we seek the best possible linear sub-space of dimension $q \leq p$ such that the following equivalent criteria are satisfied:

- Maximise the projected variance on each additional dimension we keep, *i.e,* , the first principal component should keep the maximum amount of variance possible for a single vector, the second principal component should keep the maximum amount of the *remaining* variance, up until the last component

- Minimise the reconstruction error for a reconstruction of order q

It should be intuitively clear that the equivalence holds. Keeping p principal components perfectly represents the p-dimensional data; while for any $q < p$ we seek the best rank-q approximation, which is the one that preserves the maximal amount of statistical information - given by the variability of the data, and hence the variance.

The optimal solution is found by Eigen analysis of the covariance matrix. As it is usual to subtract the mean of the data before performing PCA, notably to avoid numerical underflow, we formulate the equivalent optimisation problem in terms of the mean-centred observations.

Theorem 3.1.1. *Let* $\mathbf{S} = \frac{1}{N-1}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T}$ *with* $\tilde{X} = \mathbf{X} - \bar{\mathbf{X}}$ *the mean-centred data covariance matrix.*
The PCA basis is solution of the following constrained optimisation problem:

$$(P) \quad \max \quad \mathbf{W}^\mathsf{T}\mathbf{S}$$
$$\text{s.t} \quad \mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}$$

Given by a basis of Eigen vectors of $\mathbf{S}$*.*

A detailed discussion and proofs can be found in Bishop [15, chap. 12.1]. Finally, we give with theorem 3.1.2 a way of computing PCA without computing and diagonalising the data covariance matrix.

Theorem 3.1.2 (PCA and the SVD)**.** *A valid PCA basis can be obtain by Singular Value Decomposition of the data matrix.*

Proof. Let $\tilde{\mathbf{X}} = \mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}$, then $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T} = \mathbf{U}\mathbf{S}\mathbf{V}^\mathsf{T}\mathbf{V}\mathbf{S}^\mathsf{T}\mathbf{U}^\mathsf{T} = \mathbf{U}\mathbf{S}^2\mathbf{U}^\mathsf{T}$ with $\mathbf{S}^2$ diagonal and $\mathbf{U}$ orthogonal.
Hence, $\mathbf{U}\mathbf{S}^2\mathbf{U}^\mathsf{T}$ is a valid SVD of $\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T}$. □

Robust Principal Component Analysis

Consider a matrix $\mathbf{X}$. The Robust PCA model [21] can be described as

$$\mathbf{X} = \mathbf{A} + \mathbf{E} \tag{3.1}$$

With $\mathbf{A}$ a low-rank component and $\mathbf{E}$ a sparse matrix of arbitrarily large error terms (outliers). The goal is to recover both components separately. This problem, which seems impossible at first, can actually be tackled by solving an optimisation problem provided some assumptions on the *incoherence* and on the *sparsity* of the two signals $\mathbf{A}$ and $\mathbf{E}$ are true (Candes and Tao [22], Candes et al. [19], Candès [20]). The optimisation problem is:

$$\min \quad \text{rank}(\mathbf{A}) + ||\mathbf{E}||_0$$

With $||.||_0$ the zero pseudo-norm, *i.e,* the number of non-zero elements in the matrix. This problem is combinatorial and NP-hard; as a consequence, it is usual to seek a good approximation in the form of the convex relaxation obtained by replacing the rank and the zero pseudo-norm by their convex surrogates, namely the nuclear norm and the ℓ_1 norm:

$$\min \quad ||\mathbf{A}||_* + ||\mathbf{E}||_1$$

This problem is known as the *basis pursuit* problem [21].

Several efficient optimisation algorithms have been developed for Robust PCA, the state of the art algorithm minimises the augmented Lagrangian function through an iterative method involving proximal gradient steps for the nuclear norm and the ℓ_1 [63].

Robust 2D Principal Component Analysis

Optimisation approaches haven been proposed for Robust 2D PCA in Sun et al. [91, 92]. The method described in [91] builds on the near-optimal 2D SVD [30] an seeks a pair of orthogonal projection basis to represent a set of images.

Given a set of images $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_N\}$, bi-directional 2D-PCA approximates each image by $\mathbf{Y}_i = \mathbf{U}\mathbf{\Sigma}_i\mathbf{V}^\mathsf{T}$ with $\mathbf{U}$ and $\mathbf{V}$ orthogonal yielding the following optimisation problem:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V},\{\mathbf{\Sigma}_i\}} \quad & \tfrac{1}{N}\textstyle\sum_i ||\mathbf{Y}_i - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}||_\mathrm{F}^2 \\ \mathrm{s.t} \quad & \mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I} \\ & \mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I} \end{aligned} \tag{3.2}$$

Starting from problem 3.2 the authors make the approach robust by introducing a sparse matrix $\mathbf{E}_i$ - one for each image - to model the sparse outliers, such that the problem becomes:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V},\{\mathbf{\Sigma}_i\}} \quad & \tfrac{1}{N}\textstyle\sum_i ||\mathbf{Y}_i - \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}||_\mathrm{F}^2 + \lambda||\mathbf{E}_i||_0 \\ \mathrm{s.t} \quad & \mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I} \\ & \mathbf{V}^\mathsf{T}\mathbf{V} = \mathbf{I} \end{aligned} \tag{3.3}$$

Instead of solving a relaxed problem by replacing the ℓ_0 pseudo-norm by the ℓ_1 norm, the authors develop an alternating greedy algorithm. The two key stages are **iterative maximisation over the bases** and **alternating minimisation over $\mathbf{E}_i$ and $\mathbf{\Sigma}_i$**. In stage 1 the $\mathbf{E}_i$ and $\mathbf{\Sigma}_i$ matrices are supposed to be known and Eigen analysis is performed on the covariance matrices $\mathbf{C}_v = \frac{1}{N}\sum_i(\mathbf{Y}_i - \mathbf{E}_i)\mathbf{V}\mathbf{V}^\mathsf{T}(\mathbf{Y}_i - \mathbf{E}_i)^\mathsf{T}$ and $\mathbf{C}_u = \frac{1}{N}\sum_i(\mathbf{Y}_i - \mathbf{E}_i)\mathbf{U}\mathbf{U}^\mathsf{T}(\mathbf{Y}_i - \mathbf{E}_i)^\mathsf{T}$ to compute $\mathbf{U}$ and $\mathbf{V}$ respectively. In stage 2, the $\mathbf{\Sigma}_i$ are computed by partial differentiation, and a greedy approach is used to minimise the number of non-zero elements in $\mathrm{vec}(\mathbf{Y}_i - \mathbf{U}\mathbf{\Sigma}_i\mathbf{V}^\mathsf{T})$.

In [92], the previous approach is extended to the case of structured sparsity on the $\mathbf{E}_i$ matrices; the authors

3.1.2 Probabilistic Matrix Decompositions

Bayesian methods for robust low-rank modelling have their foundations in sparse Bayesian learning (Wipf et al. [107]) and Automatic Relevance Determination (Wipf and Nagarajan [106]). We can cite Tipping [99, 100] on the Relevance Vector Machines and Sparse Bayesian Kernel PCA as early work with durable influence on sparse Bayesian learning. These methods usually have an indirect approach to low-rank modelling, resorting to structural sparsity as a proxy to rank-minimisation. Sparsity itself is induced through the choice of suitable priors; a popular choice being to replace the intractable Laplace densities by hierarchical models involving a zero-mean Gaussian and a sparsity-promoting hyper-prior on the Gaussian distribution's precision such as the widely-used Gamma hyper-prior [99].

We briefly present in a chronological order some of the most successful methods in the field and refer to the original papers for more thorough explanations.

Standard and Bayesian Probabilistic PCA

The PPCA model is a simple generative model:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$$

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\ \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$$

With $\boldsymbol{\mu}$ the mean vector, $\mathbf{z}$ a latent variable independent of $\boldsymbol{\epsilon}$, and $\boldsymbol{\epsilon}$ an isotropic zero-mean Gaussian noise. PPCA is a restriction of the Factor Analysis model [15] since in FA, the noise covariance is only required to be diagonal. The model is fit either by maximum-likelihood or by Expectation-Maximisation (see A. P. Dempster N. M. Laird [1] for presentation of the EM algorithm). A detailed discussion can be found in Bishop [15, chap. 12.2].

Bishop [14] proposes a Bayesian treatment of the PPCA algorithm by introducing suitably chosen priors on the parameters. A variational mean field algorithm is derived.

Bilinear Probabilistic PCA

Several adaptations of Probabilistic PCA to the two-dimensional case have been proposed. Jianhua Zhao et al. [53] (2012) review the different alternatives and propose a new Bilinear PPCA model.

Some earlier models reviewed in [53] are:

- Generalized Low Rank Approximation of Matrices [GLRAM, 111]
- 2-DPCA [52]
- Probabilistic Second Order PCA [PSOPCA, 112]

Their proposed model, which they call BPPCA, is an extension of GLRAM and introduces independent column-noise and row-noise in addition to the usual white noise:

$$\begin{aligned}
&\mathbf{X} = \mathbf{C}\mathbf{Z}\mathbf{R}^\mathsf{T} + \mathbf{W} + \mathbf{C}\boldsymbol{\epsilon}_r + \boldsymbol{\epsilon}_c\mathbf{R}^\mathsf{T} + \boldsymbol{\epsilon} \\
&\mathbf{Z} \sim \mathcal{N}_{q_c,q_r}(\mathbf{0}, \mathbf{I}, \mathbf{I}),\ \boldsymbol{\epsilon}_r \sim \mathcal{N}_{q_c,d_r}(\mathbf{0}, \mathbf{I}, \sigma_r^2\mathbf{I}) \\
&\boldsymbol{\epsilon}_c \sim \mathcal{N}_{d_c,q_r}(\mathbf{0}, \sigma_c^2\mathbf{I}, \mathbf{I}),\ \boldsymbol{\epsilon} \sim \mathcal{N}_{d_c,d_r}(\mathbf{0}, \sigma_c^2\mathbf{I}, \sigma_r^2\mathbf{I})
\end{aligned}$$

With $\mathbf{W}$ the mean matrix and $\boldsymbol{\epsilon}_c, \boldsymbol{\epsilon}_r, \boldsymbol{\epsilon}$ respectively the column, row, and common noise.

Bayesian Robust Principal Component Analysis

Ding et al. [31] (2011) proposed one of the earliest Bayesian models for Robust PCA. Similarly to the non-robust models, a Bayesian treatment of RPCA requires the introduction of a white noise term, so the observation model becomes:

$$\mathbf{Y} = \mathbf{L} + \mathbf{S} + \mathbf{E}$$

Where $\mathbf{E}$ has small magnitude but dense support. $\mathbf{S}$ has large magnitude and sparse support. The authors promote sparsity by modelling the values and their supports separately:

$$\mathbf{Y} = \mathbf{D}(\mathbf{Z}\boldsymbol{\Lambda})\mathbf{W} + \mathbf{B} \circledast \mathbf{X} + \mathbf{E}$$

$\boldsymbol{\Lambda}$ is diagonal, $\mathbf{Z}$ has binary entries along the diagonal. $\mathbf{B}$ is sparse.

The low-rank component is modelled as $\mathbf{D}(\mathbf{Z}\boldsymbol{\Lambda})\mathbf{W}$. This gives a decomposition similar to SVD with the introduction of $\mathbf{Z}$ to decouple the learning of the rank and of the singular values. The rank is learnt as $||\mathbf{Z}||_0$; and $\mathbf{Z}$ serves as a mask for the $\boldsymbol{\Lambda}$. Each diagonal element is drawn from a Bernoulli trial such that $z_{kk} \sim \text{Bernoulli}(p_k)$. A Beta prior is chosen on p_k to control the desired level of sparsity. Each diagonal element of $\boldsymbol{\Lambda}$ is modelled independently by a Normal-Gamma distribution (zero-mean Gaussian with Gamma precision).

The columns of $\mathbf{D}$ and $\mathbf{W}$ are chosen independent and normally distributed:

$$\mathbf{d}_k \sim \mathcal{N}(\mathbf{0}, \frac{1}{N}\mathbf{I}_N)$$
$$\mathbf{w}_m \sim \mathcal{N}(\mathbf{0}, \frac{1}{K}\mathbf{I}_K)$$

When N and K are large, the matrices are approximately column-orthogonal, which is consistent with the analogy with the SVD. The authors report that they found imposing explicit orthogonality to be costly and unnecessary.

The sparse component is decomposed in $\mathbf{S} = \mathbf{B} \circledast \mathbf{X}$ with $\mathbf{B}$ binary. Again, this separates learning the support from learning the values such that zero components of $\mathbf{S}$ are exactly zero. As for $\mathbf{Z}$, the elements of the mask $\mathbf{B}$ have Bernoulli priors with Beta hyper-priors. The columns of $\mathbf{X}$ are drawn independently from a normal distribution with $\mathbf{0}$ mean and Gamma precision; finally the noise $\mathbf{E}$ is i.i.d Gaussian with Gamma precision.

The authors develop inference algorithms via Markov Chain Monte Carlo and Variational Bayes; as an extension, they suggest modelling the sparse outliers by a Markov Random Field to leverage spatial correlation.

Sparse Bayesian Methods for Low-Rank Matrix Estimation

Babacan et al. [6] (2012) propose a Bayesian models for probabilistic matrix completion and probabilistic Robust PCA, and derive a variational inference scheme.

Starting from an unknown matrix $X \in \mathbb{R}^{m \times n}$ of rank r, their method is based on the factorisation:

$$\mathbf{X} = \mathbf{A}\mathbf{B}^{\mathsf{T}}$$

With $\mathbf{A} \in \mathbb{R}^{m \times r}$ and $\mathbf{B} \in \mathbb{R}^{n \times r}$. They then formulate a generative model for the noisy observation $\mathbf{Y}$:

$$\mathbf{Y} = \mathbf{X} + \mathbf{E} + \mathbf{N} \tag{3.4}$$

With $\mathbf{N}$ a dense white noise matrix with small coefficients, and $\mathbf{E}$ the sparse outlier matrix with arbitrarily large magnitude.

We refer the reader to the original paper for a detailed discussion on the choice of priors and on the variational update equations.

This method has been shown to be competitive with the state of the art methods at the time of its publication, although some more powerful approaches have since been proposed. With [53], it forms the basis of the Bayesian treatment of Robust 2D PCA we develop in chapter 6.

A novel robust principal component analysis method for image and video processing

The motivation behind this method is that, in practice, the sparse noise has clustering effects so modelling using an l_p norm is a simplification that can be to restrictive for some applications. Huan et al. [51] (2016) propose a novel method based on sparse Bayesian learning principles and Markov random fields. The emphasis on the paper is put on background modelling task.

The authors propose a review of several recent Bayesian methods including [31], and [6], which we presented in this review; but also the work from Zhou et al. [120] (2013) with the *DECOLOR* algorithm dedicated to moving object detection, or Wang and Yeung [103] (2013) on *Bayesian Robust Matrix Factorization* (BRMF) and their proposed extension of the model that assumes the sparse outliers have spatio-temporal proximity (MBRMF). Zhao et al. [116] (2014) model the unknown noise by a mixture of Gaussian densities, and thus can approximate any noise distribution.

However, none of the cited methods can handle a dynamic background. The authors propose a novel methods based on the standard observation model 3.4. Both the decomposition of the low-rank matrix and the prior on the dense noise are similar to Babacan et al. [6].

To take into account the clustering effect of the sparse noise, the authors model the support of $\mathbf{S}$ by an Ising model (Binder [13], Li [61]).

Bayesian learning for Robust Principal Component Analysis

Sundin et al. [94] introduced in 2014 the *Relevance Singular Vector Machine*, named after its similarity to the work of Tipping [99]. In this 2015 paper [93], the authors investigate ways to robustify the RSVM, yielding the **rRSVM**.

Contrary to approaches such as Babacan et al. [6] who use group-sparsity as a proxy to rank-minimisation, the RSVM introduces correlation among the columns and the rows of the matrix through the factorisation:

$$\mathbf{X} = \boldsymbol{\alpha}_{\boldsymbol{L}}^{-\mathbf{1/2}}\mathbf{U}\boldsymbol{\alpha}_{\boldsymbol{R}}^{-\mathbf{1/2}} \tag{3.5}$$

$\boldsymbol{\alpha}_{\{\boldsymbol{L},\boldsymbol{R}\}}$ are positive-definite precision matrices and the elements of $\mathbf{U}$ are chosen i.i.d standard Gaussian. Therefore:

$$\mathrm{vec}(\mathbf{X}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\alpha}_R^{-1} \otimes \boldsymbol{\alpha}_C^{-1}) \tag{3.6}$$

And a low-rank is promoted by choosing Wishart (see subsection 2.4.1) priors on the precisions. The standard i.i.d Gaussian-Gamma sparse prior is chosen to model the outliers. Inference is done with EM and the method is shown to outperform several other methods.

3.2 Low-rank modelling on tensors

In this section, we provide a review of the recent tensor methods, which we will consider to be the state of the art in low-rank Tensor modelling. We chose to include both optimisation-based and Bayesian models, and will provide a comparison of the performance of some of the **robust** models over a wide range of benchmarks in Chapter 5. Some of the algorithms included in this review are dedicated to tensor completion only, and therefore are not considered robust and will not be included in the comparison. However, tensor completion - as is the case for matrix completion and matrix robust PCA - is related to tensor RPCA and based on the assumption of a low-rank tensor.

We present the different methods grouped by approach and provide the names of the original articles.

3.2.1 Matricisation-based methods

The first class of tensor decomposition that have been proposed minimises the Tucker rank by using a convex combination of the nuclear norms of the tensor unfoldings along each modes as a convex surrogate of the Tucker rank.

Tensor completion an low n-rank tensor recovery via convex optimization

Gandy et al. [38] (2011) study the problem of tensor recovery under linear measurements, and of tensor completion. By introducing a linear operator $\mathcal{A}$ such that $\mathcal{A}(\boldsymbol{\mathcal{X}}) = \mathbf{b}$, or $\boldsymbol{\mathcal{X}}_\Omega = \boldsymbol{\mathcal{T}}_\Omega$ if $\mathcal{A}$ denotes the projection on the set of tensors with observed entries Ω, we obtain the problem of *low n-rank tensor pursuit*:

$$\begin{aligned} \min \quad & \textstyle\sum_i ||\mathbf{X}_{[i]}||_* \\ \text{s.t} \quad & \mathcal{A}(\boldsymbol{\mathcal{X}}) = \mathbf{b} \end{aligned}$$

They propose a variant to take in account the presence of noise by replacing the equality constraint by an inequality $||\boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{X}}) - \mathbf{b}||_2 \leq \epsilon$. An unconstrained Lagrangian formulation is also proposed as:

$$\min \sum_i ||\mathbf{X}_{[i]}||_* + \frac{\lambda}{2}||\mathcal{A}(\boldsymbol{\mathcal{X}}) - \mathbf{b}||_2^2$$

Robust low-rank tensor recovery: models and algorithms

Goldfarb and Qin [40] (2014) introduce a range of both convex and non-convex algorithms for low-rank modelling based on the Tucker rank. Introducing $\mathrm{CTrank}(\boldsymbol{\mathcal{X}})$ to denote the chosen convex surrogate for the Tucker rank, the authors investigate two approaches: the *singleton* model, and the *mixture* model.

In the singleton model, $\mathrm{CTrank}(\boldsymbol{\mathcal{X}}) = \sum_i ||\mathbf{X}_{[i]}||_*$. The authors define *Higher Order RPCA (HoRPCA)* for the singleton model (HORPCA-S) by:

$$\begin{aligned} \min \quad & \textstyle\sum_i ||\mathbf{X}_{[i]}||_* + \lambda_1||\boldsymbol{\mathcal{E}}||_1 \\ \text{s.t} \quad & \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{B}} \end{aligned}$$

With $\boldsymbol{\mathcal{B}}$ the observed tensor, $\boldsymbol{\mathcal{X}}$ the low-rank tensor, and $\boldsymbol{\mathcal{E}}$ the sparse outliers. The problem can be solved with variable splitting; introducing one $\boldsymbol{\mathcal{X}}_i$ per mode:

$$\begin{aligned} \min \quad & \textstyle\sum_i ||\mathbf{X}_{[i]}^{(i)}||_* + \lambda_1||\boldsymbol{\mathcal{E}}||_1 \\ \text{s.t} \quad & \forall i,\ \boldsymbol{\mathcal{X}}_i + \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{B}} \end{aligned}$$

The singleton model is included in our comparative study. In the mixture model, we only require that the low-rank tensor be the sum of component tensors each low-rank in the corresponding mode. The constraint of the problem becomes $\sum_i \boldsymbol{\mathcal{X}}_i + \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{B}}$.

Even though the ADMM approach they propose is shown to be faster and more accurate, the authors also denote that unconstrained - Lagrangian - formulations of both the singleton and mixture models can be solved via accelerated proximal gradient methods (FISTA).

Robust Tensor Decomposition with Gross Corruption

Consider the Robust PCA-type observation model.

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{W}}^* + \boldsymbol{\mathcal{V}}^* + \boldsymbol{\mathcal{E}}$$

With $\boldsymbol{\mathcal{W}}^*$ a low-rank tensor, $\boldsymbol{\mathcal{V}}^*$ a sparse corruption tensor, and $\boldsymbol{\mathcal{E}}$ a dense noise tensor whose elements are assumed to be i.i.d Gaussian with zero mean and bounded variance σ^2.

Gu et al. [43] (2014) introduce an unconstrained model where the constraint is replaced by a squared loss:

$$\min ||\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{V}}||_2^2 + \lambda_M \frac{1}{N} \sum_n ||\mathbf{W}_{[n]}||_* + \mu_M||\boldsymbol{\mathcal{V}}||_1$$

Robust Low-Rank Modelling on Matrices and Tensors

In his 2014 MSc Thesis, Papamakarios [76] develops matricisation-based methods for robust low-rank modelling on tensors. We refer the reader to the original manuscript for more details, and will only present briefly two methods we chose to include in our experimental comparison.

Tensor RPCA is defined in a similar fashion as for HORPCA-S [40], the sum of the nuclear norms of the mod-n matricisations is taken as a convex surrogate to the Tucker rank and the problem is solved by ADMM or APG. *Tensor BRPCA* is a similar model but avoids computing expensive SVDs on the unfoldings by using the variational characterisation of the matrix nuclear norm given in theorem 2.1.15.

Robust low-rank tensor recovery with redescending M-estimator

Under Gaussian noise, the least-square penalty recovers the maximum likelihood estimate. M-estimators are robust statistical estimators and define loss functions suitable for the recovery of a low-rank tensor contaminated with non-Gaussian noise or outliers.

Yang et al. [110] (2015b) propose the use of Welsh and Cauchy losses. The data-fitting risk takes the form:

$$J_\sigma(\sum(\boldsymbol{\mathcal{X}})) = \sum_i \rho_\sigma(\langle \boldsymbol{\mathcal{A}}_i, \sum \boldsymbol{\mathcal{X}} \rangle - b_i)$$

Here $\boldsymbol{\mathcal{X}}$ denotes a tuple of tensors $\boldsymbol{\mathcal{X}} = (\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_N)$. By letting $\sum \boldsymbol{\mathcal{X}} = \sum_i \boldsymbol{\mathcal{X}}_i$ they define a mixture model similar to that of Goldfarb and Qin [40]. Denoting by $\mathbf{X}_{[j]}^{(j)}$ the mode-j unfolding of the j^{th} component tensor of the mixture, the model takes the form:

$$\min_{\boldsymbol{\mathcal{X}} \in \mathbb{T}^N} J_\sigma(\sum_j \boldsymbol{\mathcal{X}}_j) + \lambda \sum_j ||\mathbf{X}_{[j]}^{(j)}||_*$$

The authors derive proximal and linearised block coordinate descent methods and show the approaches satisfy the Kurdyka-Lojasiewicz property, which establishes the global convergence of the algorithms.

If the rank is known in advance the authors also define the following rank-constrained problem:

$$\min_{\boldsymbol{\mathcal{X}} \in \mathbb{T}^N} J_\sigma(\sum_j \boldsymbol{\mathcal{X}}_j) \quad \text{s.t} \quad \text{rank}(\mathbf{X}_{[j]}^{(j)}) \leq R_j \forall j$$

The constraints can be relaxed by defining the set M_{R_j} of suitably sized matrices of rank at most R_j and the penalty function $\delta_C(x) = 0$ if $x \in C$, $+\infty$ otherwise. The objective becomes:

$$\min_{\boldsymbol{\mathcal{X}} \in \mathbb{T}^N} J_\sigma(\sum_j \boldsymbol{\mathcal{X}}_j) + \sum_j \delta_{M_{R_j}}(\mathbf{X}_{[j]}^{(j)})$$

We included both the Welsh and Cauchy losses with soft-thresholding

3.2.2 Robust Factorisations

While matricisation-based methods operate by regularisation on the original tensor, robust tensor factorisations have been proposed that seek the best decomposition of a given form. In this section, we review some of the recent developments based on the CP and Tucker decompositions.

Factor Matrix trace Norm Minimization for Low-Rank Tensor Completion

This 2014 method from Liu et al. [65] is based on Theorem 3.2.1:

Theorem 3.2.1. *Let $\boldsymbol{\mathcal{X}} = \mathbf{U}^{(1)} \circ \dots \mathbf{U}^{(N)}$ be the CP decomposition of a tensor $\boldsymbol{\mathcal{X}}$ and $\mathbf{X}_{[i]}$ the mode-i unfolding. Then:*

$$\forall n, \operatorname{rank}(\mathbf{X}_{[n]}) \leq \operatorname{rank}(\mathbf{U}^{(n)})$$

Relying on the above result, the authors find an upper bound of the Tucker rank of $\boldsymbol{\mathcal{X}}$. The model is:

$$\begin{aligned} \min \quad & \textstyle\sum_n w_n \operatorname{rank}(\mathbf{U}^{(n)}) \\ \text{s.t} \quad & \boldsymbol{\mathcal{X}} = \mathbf{U}^{(1)} \circ \dots \mathbf{U}^{(N)} \\ & \boldsymbol{\mathcal{X}}_\Omega = \boldsymbol{\mathcal{T}}_\Omega \end{aligned}$$

Which the authors relax in:

$$\begin{aligned} \min \quad & \textstyle\sum_n w_n ||\mathbf{U}^{(n)}||_* + \frac{\lambda}{2} ||\boldsymbol{\mathcal{X}} - \mathbf{U}^{(1)} \circ \dots \mathbf{U}^{(N)}||_{\mathrm{F}}^2 \\ \text{s.t} \quad & \boldsymbol{\mathcal{X}}_\Omega = \boldsymbol{\mathcal{T}}_\Omega \end{aligned}$$

And solve via ADMM with splitting.

Generalized Higher-Order Tensor Decomposition via Parallel ADMM

Shang et al. [87] (2014) develop parallel algorithms for the problem of approximating a low Tucker-rank tensor $\boldsymbol{\mathcal{T}}$ with a tensor $\boldsymbol{\mathcal{X}}$:

$$\min \sum_n ||\mathbf{X}_{[n]}||_* + \frac{\lambda}{2} ||\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{T}}||_{\mathrm{F}}^2$$

Similar to approaches described in subsection 3.2.1. However, the authors also study instances of rank-minimization for the Tucker decomposition of $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}} \times_{i=1}^{N} \mathbf{U}^{(i)}$:

$$\begin{aligned} \min \quad & \textstyle\sum_n ||\mathbf{G}_{[n]}||_* + \frac{\lambda}{2} ||\boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{G}} \times_{i=1}^{N} \mathbf{U}^{(i)}||_{\mathrm{F}}^2 \\ \text{s.t} \quad & \forall i,\ {\mathbf{U}^{(i)}}^{\mathsf{T}} \mathbf{U}^{(i)} = I \end{aligned}$$

Even though the problems described aren't instances of robust modelling or tensor completion, we chose to give this brief description of the paper as it is an interesting model developed for the Tucker/HOSVD decomposition.

Tucker factorization with missing data with application to low-n-rank tensor completion

Filipović and Jukić [36] (2015) propose a Tucker decomposition with missing values.

Let $\boldsymbol{\mathcal{W}}$ be an indicator tensor, and $\boldsymbol{\mathcal{X}}$ the latent (unobserved) tensor. The method fits the best Tucker factorisation using only the observed elements $\boldsymbol{\mathcal{X}}_\Omega$. The problem is presented as the *unconstrained* minimisation of the following objective:

$$f_{\boldsymbol{\mathcal{W}}}(\boldsymbol{\mathcal{G}}, \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)}) = ||\boldsymbol{\mathcal{W}} \circledast (\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{G}} \times_{i=1}^{N} \mathbf{A}^{(i)}||_{\mathrm{F}}^2$$

The authors suggest adding constraints could be possible, to enforce, *e.g,* non-negativity or orthogonality.

Low-rank tensor recovery via iterative hard thresholding

Rauhut et al. [82] (2016) introduce a version of the iterative hard thresholding algorithm for the higher order singular value decomposition to solve problems of the form

$$\min \operatorname{rank}(\boldsymbol{\mathcal{X}}) \quad \text{s.t} \quad y = \boldsymbol{\mathcal{A}}(\boldsymbol{\mathcal{X}})$$

Similar to the matrix hard thresholding operator, which computes the best rank-k approximation of a matrix by keeping only the first k components of the SVD, they define a tensor hard thresholding operator $\boldsymbol{\mathcal{H}}_r$ that restricts the HOSVD components to a given Tucker rank $r = (r_1, \ldots, r_d)$.

Robust Low-Rank Modelling on Matrices and Tensors

In his 2014 MSc Thesis, Papamakarios [76] investigates robust versions of both the CP and the Tucker decompositions; the optimisation problems are solved by convex optimisation with either splitting or linearisation.

We refer to the original manuscript for more details. Both decompositions with splitting are included in our experimental study.

3.2.3 Frontal slices

Some tensor methods have been proposed that are explicitly designed to decompose 3-way tensors based on their frontal slices. Such methods find applications mainly in computer vision where it is common to represent a set of images, like a video sequence, by a 3-way tensor. In this case, the frontal slices are the main objects of interest.

Historical approaches

In **Robust tensor factorization using R_1 norm** (2008), Ding [29] first makes robust a standard Matrix PCA objective by replacing the ℓ_2 (Frobenius) loss function by a more robust loss - in this case the R_1 norm. The sparse outlier component is not separated from the rest; only the low-rank factor is recovered explicitly. Let us first define the matrix R_1 norm.

Definition 3.2.2 (Matrix R_1 norm)
Let $\mathbf{X}$ *be a matrix, formed from the column vectors* $[\mathbf{x}_1, \ldots, \mathbf{x}_N]$. *The* R_1 *norm of* $\mathbf{X}$ *is defined as:*

$$||\mathbf{X}||_{R_1} = \sum_{i=1}^{N} \left(\sum_{j=1}^{d} x_{ji}^2 \right)^{1/2}$$

Recall that for the same matrix, the Frobenius norm is $||\mathbf{X}||_{\mathrm{F}} = \left(\sum_{i=1}^{N} \sum_{j=1}^{d} x_{ji}^2 \right)^{1/2}$.
We get the modified objective of the PCA problem:

$$\min_{\mathbf{U},\mathbf{V}} ||\mathbf{X} - \mathbf{U}\mathbf{V}||_{R_1} = \sum_{i=1}^{N} ||x_i - \mathbf{U}v_i||$$

The 3-way tensor version is based on the 2D-SVD and is defined as:

$$\min_{\mathbf{L},\mathbf{R},\{\mathbf{M}_i\}} \sum_i \sqrt{||\mathbf{X}_i - \mathbf{L}\mathbf{M}_i\mathbf{R}^\mathsf{T}||_{\mathrm{F}}^2}$$

And aims at recovering a pair of bases $\mathbf{L}$, $\mathbf{R}$ and a set of matrices $\{\mathbf{M}_1, \ldots, \mathbf{M}_N\}$ such that the modified PCA objective is minimal.

The authors actually provide an extension to the D-way tensor case, and propose a decomposition that introduces $D-1$ subspaces. For example in the 4-way case, given a set $\boldsymbol{\mathcal{Y}}$ or 3-way tensors $\boldsymbol{\mathcal{Y}}_l$, the method consists in solving the following problem:

$$\begin{aligned} \min \quad & \textstyle\sum_l ||\boldsymbol{\mathcal{Y}}_l - \mathbf{U} \otimes_1 \mathbf{V} \otimes_2 \mathbf{W} \otimes_3 \mathbf{M}_l||_{R_1} \\ \text{s.t} \quad & \mathbf{V}^\mathsf{T}\mathbf{V} = I, \mathbf{U}^\mathsf{T}\mathbf{U} = I, \mathbf{W}^\mathsf{T}\mathbf{W} = I \end{aligned}$$

More recently, Zhang and Wang [114] (2013) propose in **Simultaneous rectification and alignment via robust recovery of low-rank tensors** a method which considers an observation tensor $\boldsymbol{\mathcal{A}}$ and aims at finding a low-rank component $\boldsymbol{\mathcal{L}}$ and a sparse outliers term $\boldsymbol{\mathcal{E}}$ such that:

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}$$

Their approach addresses a problem commonly encountered in real datasets, which is that the data are required to be well-aligned; a slight misalignment can break the low-rank structure and render low-rank modelling methods ineffective.

The authors adopt a set of transformations $\Gamma = \{\boldsymbol{\tau}_1^{-1}, \ldots, \boldsymbol{\tau}_{I_3}^{-1}\}$ - one for each frontal slice of $\boldsymbol{\mathcal{A}}$ - so that the model becomes:

$$\boldsymbol{\mathcal{A}} \star \Gamma = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}$$

Where $\star$ denotes applying each transformation to the corresponding slice.

They then describe a convex-optimisation strategy to minimise the *Tucker* rank of $\boldsymbol{\mathcal{L}}$. Choosing a convex combination $\sum_i \alpha_i ||\mathbf{L}_{[i]}||_*$ of the nuclear norm of the unfoldings of $\boldsymbol{\mathcal{L}}$ as a convex surrogate for the rank, the authors derive an ADMM algorithm with variable splitting on each $\mathbf{L}_{[i]}$ and *linearise* the highly non-linear constraint $\boldsymbol{\mathcal{A}} \star \Gamma = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}$ as follows:

$$\boldsymbol{\mathcal{A}} \star \Gamma + \text{fold}_3 \left((\sum_i J_i \Delta\Gamma \epsilon_i \epsilon_i^\mathsf{T})^\mathsf{T} \right) = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}$$

Tensor Nuclear Norm in the Fourier domain

A recent class of methods that operate on the frontal slices are based on the notions of tensor t-product, t-SVD, and nuclear norm described in subsections 2.2.4 and 2.2.5.

Novel methods for multilinear data completion and de-noising based on tensor-SVD (2014) from Zhang et al. [115] introduces two optimisation problems for tensor completion, and tensor RPCA:

$$\begin{aligned} \min \quad & ||\boldsymbol{\mathcal{L}}||_{\text{TNN}} \\ \text{s.t} \quad & P_\Omega(\boldsymbol{\mathcal{X}}) = P_\Omega(\boldsymbol{\mathcal{L}}) \end{aligned} \tag{3.7}$$

$$\begin{aligned} \min \quad & ||\boldsymbol{\mathcal{L}}||_{\text{TNN}} + \lambda ||\boldsymbol{\mathcal{E}}||_{1,1,2} \\ \text{s.t} \quad & \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}} \end{aligned} \tag{3.8}$$

With $||\boldsymbol{\mathcal{E}}||_{1,1,2} = \sum_{i,j} ||\boldsymbol{\mathcal{E}}(i,j,:)||_{\text{F}}$ in Matlab notation.

According to the paper, for $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times 3}$, a theoretically optimal choice for λ is $\lambda^* = \frac{1}{\max(I_1, I_2)}$. This algorithm is included in our comparative study.

Hu et al. [50] (2015) minimise the *twist tensor Nuclear Norm*; a variant of the tensor nuclear norm obtained from the twist tensor:

$$\text{t} - \text{TNN}(\boldsymbol{\mathcal{X}}) = ||\boldsymbol{\mathcal{X}}||_{\vec{*}} = ||\vec{\boldsymbol{\mathcal{X}}}||_* = ||\text{bcirc}(\vec{\boldsymbol{\mathcal{X}}})||_*$$

The twist tensor is a rearrangement of the frontal slices of $\boldsymbol{\mathcal{X}}$ so that not only the time correlation is preserved when matricising the tensor. In $\mathbf{X}_{[3]}$ the time correlation is preserved since every matricised frame is aligned with the other, but the spacial information is lost (if you remember, we talked about something similar the last time we met or the time before). The twist tensor aims at solving this by exploiting the structure of block circulant matrices, so that the block-circulant matrix of the frontal slices preserves both the temporal and spatial information. The optimisation problem is dedicated to tensor completion and is solved via ADMM.

$$\begin{aligned} \min \quad & ||\boldsymbol{\mathcal{X}}||_{\vec{*}} \\ \text{s.t} \quad & P_\Omega(\boldsymbol{\mathcal{X}}) = P_\Omega(\boldsymbol{\mathcal{X}}_0) \end{aligned}$$

Finally, the very recent **Tensor Robust PCA with exact recovery** by Lu et al. [66] (CVPR 2016) builds on Zhang et al. [115]. Assuming $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}}_0 + \boldsymbol{\mathcal{E}}_0$, the model finds the best approximating $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{E}}$ by solving the following *Tensor RPCA* problem:

$$\begin{aligned} \min \quad & ||\boldsymbol{\mathcal{L}}||_* + \lambda||\boldsymbol{\mathcal{E}}||_1 \\ \text{s.t} \quad & \boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}} \end{aligned}$$

Where $||.||_*$ is a variant of the Tensor Nuclear Norm defined in relationship with the t-SVD (2.2.4). The method minimises the *tensor average-rank*.

3.2.4 Other optimisation-based approaches

So far, we have reviewed convex optimisation methods for:

- Regularising the tensor unfoldings
- Finding the best approximating factorisation of a given form
- Optimising functions of the tensor frontal slices

In this section, we investigate alternative methods for low-rank modelling. The first family of methods we present perform *non-convex* optimisation, while the second indirectly minimises the tensor CP-rank through duality.

Non-convex optimisation

Anandkumar et al. [4] (2016) propose a tensor extension of non-convex RPCA known as **Non-convex Tensor Robust PCA**. The paper is about recovering the low-rank and sparse components of tensors in an instance of the CP decomposition:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{L}}^* + \boldsymbol{\mathcal{S}}^*, \quad \boldsymbol{\mathcal{L}}^* = \sum_{i=1}^{r} u_i \otimes u_i \otimes u_i$$

The algorithm is in two phases:

- Update the low-rank estimate $\hat{\boldsymbol{\mathcal{L}}}$ through eigenvector computation of $\boldsymbol{\mathcal{T}} - \hat{\boldsymbol{\mathcal{S}}}$ via a modification of the tensor power method
- Update the sparse component through hard thresholding of the residual $\boldsymbol{\mathcal{T}} - \hat{\boldsymbol{\mathcal{L}}}$

We refer the reader to the original manuscript for details.

Several algorithms have been adapted to incorporate information about the true hidden rank through non-convex optimisation via **Hard-thresholding**.

In the case of an unknown rank the standard approach is to replace the rank function by a suitable convex surrogate and to optimise the objective to find the minimum rank solution. The rank of the solution can often be used as a good - if not exact - indication of the true rank. When the rank or a good upper bound on the rank is known, either through expert knowledge or by first applying a convex method to the problem, the information about the true rank can be incorporated in the optimisation problem as a constraint.

Goldfarb and Qin [40] (2014) propose a non-convex version of their HORPCA algorithm (see 3.2.1). The optimisation problem minimises $||\boldsymbol{\mathcal{E}}||_1$ with constraints $\boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{B}}$ and $\forall i, \text{rank}(\mathbf{X}_{[i]}) \leq r_i$:

$$\begin{aligned} \min \quad & ||\boldsymbol{\mathcal{E}}||_1 \\ \text{s.t} \quad & \boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{E}} = \boldsymbol{\mathcal{B}} \\ & \forall i, \text{rank}(\mathbf{X}_{[i]}) \leq r_i \end{aligned}$$

The rank constraint can then be relaxed in the objective by introducing a suitable indicator function, and the problem can be solved by applying *hard thresholding* at each iteration.

Spectral norm minimisation

In their 2015a paper **A rank-one tensor updating algorithm for tensor completion**, Yang et al. [109] devise a rank-one updating algorithm for tensor completion.

The method optimises the tensor nuclear norm as defined in Definition 2.2.23 via the conditional gradient method. Recall the tensor nuclear norm is defined as:

$$||\boldsymbol{\mathcal{X}}||_* = \inf\{\sum_{p=1}^{+\infty} \sigma_p | \boldsymbol{\mathcal{X}} = \sum_{p=1}^{+\infty} \sigma_p \boldsymbol{\mathcal{X}}_p, \sigma_p \geq 0\}$$

Where $\boldsymbol{\mathcal{X}}_p$ are rank-one tensors of unit Frobenius norm, and that it is dual to the tensor spectral norm defined as:

$$||\boldsymbol{\mathcal{X}}||_2 = \max_{\forall i,\ ||y_i||=1} \langle \boldsymbol{\mathcal{X}}, y_1 \otimes \ldots \otimes y_N \rangle$$

Computing the tensor nuclear norm is NP-hard and difficult to approximate. Finding the spectral norm is also NP-hard, but all NP-hard problems are not equally difficult and various approximation methods exist for the tensor spectral norm, so the problem is much more tractable in practice. The method proposed in [109] optimises the tensor spectral norm as a proxy to the tensor nuclear norm.

In **Scalable and sound low-rank tensor learning** Cheng et al. [26] (2016) very recently proposed a spectral norm-based approach for the problem of learning a low-rank tensor $\boldsymbol{\mathcal{W}}$ that best approximates $\boldsymbol{\mathcal{X}}$:

$$\min_{\boldsymbol{\mathcal{W}}} l(\boldsymbol{\mathcal{W}}) + \lambda \text{rank}(\boldsymbol{\mathcal{W}})$$

With l a measure of discrepancy between $\boldsymbol{\mathcal{W}}$ and $\boldsymbol{\mathcal{X}}$.

The authors' goal is to avoid resorting to approximating the tensor nuclear norm by the sum of the nuclear norms of the unfoldings, as this breaks the tensor's structure. Instead, and similarly to Yang et al. [109], they choose the tensor nuclear norm defined in Definition 2.2.23 as their convex surrogate for the rank, and solve the dual problem with the tensor spectral norm.

3.2.5 Bayesian methods

Bayesian methods for low-rank and robust low-rank modelling on tensor have been proposed recently. All the presented methods share the characteristic that they are Bayesian CP models.

Bayesian Robust Tensor Factorization for Incomplete Multiway Data

Zhao et al. [118] (2016) propose a Robust Bayesian CP factorization for Tensor RPCA with approximate Bayesian inference in the Variational Bayes framework. They demonstrate the effectiveness of the method on artificial data and for background subtraction tasks. The method can be seen as an extension of a non-robust Bayesian CP factorization described by the same authors in Zhao et al. [117]. This algorithm is part of our comparative study.

Robust tensor factorization with unknown noise

Chen et al. [25] (2016) describe a Bayesian CP decomposition dedicated to image denoising and devised to handle complex forms of noise by approximating the unknown noise distribution with a mixture of Gaussians (MoG) fitted by Expectation-Maximization (we refer to Bishop [15], Hastie et al. [48], Murphy [73] for a thorough discussion on EM and mixture models).

This method is a tensor version of *Robust principal component analysis with complex noise* [116] (2014) from the same authors.

Bayesian factorizations of big sparse tensors

In this 2014 article from Zhou et al. [119], the authors develop a Bayesian sparse PARAFAC model for very large tensors.
Recall the standard CP factorization of a p-mode tensor is:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{U_r}^{(1)} \circ \dots \mathbf{U_r}^{(p)} \tag{3.9}$$

The idea of sparse PARAFAC (sp-PARAFAC) is as follows:

For a given r as it appears in equation (3.9) , partition the modes into two mutually exclusive sets $S_r \cup S_r^{\perp}$. The sp-PARAFAC factorisation is:

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} \mathbf{U_r}^{(1)} \circ \dots \mathbf{U_r}^{(p)}, \quad \mathbf{U_r}^{(j)} = \mathbf{U_0}^{(j)} \text{ for } j \in S_r^{\perp}$$

Therefore instead of enforcing blocks of exact zeros in $\boldsymbol{\mathcal{X}}$ the model reduces the number of degrees of freedom used in some directions by setting a large fraction of the $\mathbf{U_r}^{(j)}$ to a *baseline factor* $\mathbf{U_0}^{(j)}$. By including j in $S_r^{\perp}$ for some r only, one can use fewer degrees of freedom in characterising the interaction between the j^{th} factor and the other factors. In practice, we will learn the set of $\{S_r\}$ using a Bayesian approach, as the appropriate lower dimensional structure is typically not known in advance.

Bayesian Poisson Tensor Factorization for Inferring Multilateral Relations from Sparse Dyadic Event Counts

This model proposed by Schein et al. [86] (2015) is tailored for applications to dynamic pairwise interaction patterns. The Bayesian tensor factorisation model infers latent group structures from the data.

An example of dyadic event is: "country i took action a toward country j at time t. As the events are represented as a tensor of counts, a Poisson model is natural. The authors develop a Bayesian Poisson tensor factorisation to infer a **low-dimensional**, interpretable representation of their salient patterns and perform approximate inference via VB.

We give the example of a four-way model and refer the reader to the original paper for details.

$$y_{ijat} \approx \hat{y}_{ijat} = \sum_{k=1}^{K} \theta_{ik}^{(1)} \theta_{jk}^{(2)} \theta_{ak}^{(3)} \theta_{tk}^{(4)}$$

It its interesting to note that if $y_{ijat} \sim \mathrm{Pois}(y_{ijat}; \hat{y}_{ijat})$ then the process is known as Poisson factorisation and can be performed by MLE of the $\theta^{(i)}$. The Bayesian approach involves imposing priors on the latent parameters (usually Gamma to preserve conjugacy with the Poisson likelihood).

Chapter 4

Novel algorithms for Robust 2D PCA

4.1 Introduction

In this chapter, we study a class of convex optimisation algorithms for low-rank modelling on 3-way tensors based on a decomposition of the *frontal slices* of the tensor. In computer vision applications, 3-way tensors are natural representations of image collections, such as: face images of the same person under various lighting conditions, video sequences, or multispectral images where each slice represents one particular subset of the spectrum.

Considering an image collection represented by a 3-way tensor $\boldsymbol{\mathcal{X}}$, our aim is to efficiently model the spacial correlation within and between images by seeking a pair of optimal matrices $\mathbf{U_c}$ and $\mathbf{U_r}$ that represent common low-dimensional column and row subspaces. The specific information about each image is stored as the frontal slices of a low-dimensional 3-way tensor $\boldsymbol{\mathcal{T}}$ so that the low-rank component of a given matrix $\mathbf{X}_n$ is $\mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T}$. A sparse outlier tensor $\boldsymbol{\mathcal{E}}$ is also introduced, with each frontal slice of $\boldsymbol{\mathcal{E}}$ containing the outliers extracted from $\mathbf{X}_n$.

As a proxy to rank-minimisation, we minimise well-chosen properties of the matrices $\mathbf{U_c}$ and $\mathbf{U_r}$. We study different approaches to the problem and demonstrate experimentally that our methods effectively recover signals that match the model. We also discuss the complexity and scalability of the algorithms and the possible implementation choices. An efficient and modular parallel implementation is made available as a Matlab package on the author's Github page. We conclude by highlighting relations between our models and dictionary learning.

4.2 Generic form of the optimisation problem

We aim at minimising the rank of the recovered component while maximising the sparsity of the outlier component. The most generic form of the optimisation problems we solve can be described as:

$$\begin{aligned} \min \quad & f(\mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}}) \\ \text{s.t} \quad & \forall n, \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} + \mathbf{E}_n \end{aligned}$$

More specifically, we will study problems of the following more restrictive form:

$$\begin{aligned} \min \quad & \alpha_c f(\mathbf{U_c}) + \alpha_r f(\mathbf{U_r}) + g(\boldsymbol{\mathcal{T}}) + \Omega(\boldsymbol{\mathcal{E}}) \\ \text{s.t} \quad & \forall n, \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} + \mathbf{E}_n \end{aligned} \tag{4.1}$$

f, g can, but need not, be smooth functions while Ω is chosen to be a non-smooth sparsity promoting penalty. Ω can be designed to incorporate structural information about the noise, but due to time

constraints we will limit our study to the case of the sum of the ℓ_1-norm of the frontal slices of $\boldsymbol{\mathcal{E}}$. Similarly, we will find convenient to consider penalties on $\boldsymbol{\mathcal{T}}$ that are expressed in terms of the frontal slices of tensor. Therefore, the final family of problems we consider is:

$$\begin{aligned} \min \quad & \alpha_c f(\mathbf{U_c}) + \alpha_r f(\mathbf{U_r}) + \textstyle\sum_n g(\mathbf{T}_n) + \sum_n \Omega(\mathbf{E}_n) \\ \text{s.t} \quad & \forall n, \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} + \mathbf{E}_n \end{aligned} \tag{4.2}$$

We chose separable penalties on $\mathbf{U_c}$, $\mathbf{U_r}$, and $\mathbf{T}$ to achieve tractability. The case where the penalties are non-smooth will be treated through *operator splitting* by introducing additional additional variables and their associated equality constraints. The most simple choice of penalties is $f(\mathbf{U_c}) = \frac{\alpha_c}{2}||\mathbf{U_c}||_\mathrm{F}^2$, $f(\mathbf{U_r}) = \frac{\alpha_r}{2}||\mathbf{U_r}||_\mathrm{F}^2$ and $h(\boldsymbol{\mathcal{T}}) = \frac{\alpha_t}{2}\sum_n ||\mathbf{T}_n||_\mathrm{F}^2$ and will be shown to be equivalent to Tikhonov regularisation on a set of least squares sub-problems.

Other penalties we studied aim at introducing more sparsity in the solution. If we choose f and g to be weighted ℓ_1/ℓ_2 norms we obtain *non-overlapping Group Lasso* problems that we solve by splitting. Similarly, we can choose to replace the Frobenius norm on the frontal slices of $\boldsymbol{\mathcal{T}}$ by the ℓ_1 norm.

4.3 Non-orthogonal Robust 2D PCA

The method we present in this section is at the basis of all our other developments, including the Bayesian treatment of Chapter 6.

4.3.1 Optimisation problem

We solve the following non-linear constrained problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\left(\alpha_c||\mathbf{U_c}||_\mathrm{F}^2 + \alpha_r||\mathbf{U_r}||_\mathrm{F}^2 + \alpha_t\textstyle\sum_n ||\mathbf{T}_n||_\mathrm{F}^2\right) + \lambda\sum_n ||\mathbf{E}_n||_1 \\ \text{s.t} \quad & \forall n,\ \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} + \mathbf{E}_n \end{aligned}$$

The problem is non-linear, non-smooth and jointly non-convex. In order to solve it efficiently, we propose an Alternating Directions of Multipliers Method with a proximal step for estimating $\boldsymbol{\mathcal{E}}$. This makes solving the problem tractable as, by treating each variable independently, each sub-problem is convex and easy to solve. We denote the n^{th} constraint by h_n.

4.3.2 Augmented Lagrangian

We define the augmented Lagrangian function $L(\mathbf{U_c}, \mathbf{U_r}, \mathbf{T}, \mathbf{E}, \lambda, \alpha_c, \alpha_r, \mu, \mathbf{\Lambda})$:

$$\begin{aligned} L(\mathbf{U_c}, \mathbf{U_r}, \mathbf{T}, \mathbf{E}, \lambda, \mu, \mathbf{\Lambda}) = \frac{\alpha_c}{2}||\mathbf{U_c}||_\mathrm{F}^2 + \frac{\alpha_r}{2}||\mathbf{U_r}||_\mathrm{F}^2 + \frac{\alpha_t}{2}\sum_n ||\mathbf{T}_n||_\mathrm{F}^2 + \lambda\sum_n ||\mathbf{E}_n||_1 + \\ \sum_n \langle \mathbf{\Lambda}_n, \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n\rangle + \\ \frac{1}{2}\mu||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2 \end{aligned}$$

- λ is a regularization parameter that controls the weight of the 1-norm of the sparse errors.
- α_c, α_r, and α_t are regularization parameters that control the weight of the Frobenius penalties on $\mathbf{U_c}$, $\mathbf{U_r}$, and each $\mathbf{T}_n$.

- $\mathbf{\Lambda}_n$ is the matrix of Lagrange multipliers associated with h_n.
- μ controls the weight of the penalty term (squared Frobenius norm of h_n) and is simultaneously the step size for the dual variables.

4.3.3 Update equations and algorithm

We can now derive the update equations and the whole algorithm including initialisation.

Sparse error

The ℓ_1-norm penalty is non-smooth, finding the value $\mathbf{E}_n^*$ that minimises L involves the proximal mapping of the ℓ_1 norm, namely the soft-shrinkage operator $\mathcal{S}_{\lambda/\mu}$:

$$\mathbf{E}_n = \mathcal{S}_{\lambda/\mu}(\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} + \frac{1}{\mu}\mathbf{\Lambda}_n)$$

To show this, it suffices to merge the linear term (*i.e,* the dot product of the Lagrange multiplier with the constraint) with the squared Frobenius penalty by observing that the additional terms obtained when expanding the new penalty are constant with respect to $\mathbf{E}_n$ and hence do not affect the value of the arg-minimum of the expression.

Proof.

$$\begin{aligned}
\mathbf{E}_n{}^* &= \operatorname*{argmin}_{\mathbf{E}_n} \lambda||\mathbf{E}_n||_1 + \frac{\mu}{2}||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n + \frac{1}{\mu}\mathbf{\Lambda}_n||_\mathrm{F}^2 \\
&= \operatorname*{argmin}_{\mathbf{E}_n} \lambda||\mathbf{E}_n||_1 + \frac{\mu}{2}\langle \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n + \frac{1}{\mu}\mathbf{\Lambda}_n,\, \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n + \frac{1}{\mu}\mathbf{\Lambda}_n\rangle \\
&= \operatorname*{argmin}_{\mathbf{E}_n} \lambda||\mathbf{E}_n||_1 + \frac{\mu}{2}\Big[||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2 + 2\langle\frac{1}{\mu}\mathbf{\Lambda}_n,\, \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n\rangle \\
&+ \frac{1}{\mu^2}||\mathbf{\Lambda}_n||_\mathrm{F}^2\Big] \\
&= \operatorname*{argmin}_{\mathbf{E}_n} \lambda||\mathbf{E}_n||_1 + \frac{\mu}{2}||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2 + \langle\mathbf{\Lambda}_n,\, \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n\rangle
\end{aligned}$$

□

The soft-shrinkage operator has a close-form expression, as shown in Property 4.3.1:

Property 4.3.1. *Let* $\mathbf{x} \in \mathbb{R}^n$ *and* $i \in \{1, \ldots, n\}$*,*

$$\mathrm{prox}_{\lambda||.||_1}(\mathbf{x})_i = \mathcal{S}_\lambda(\mathbf{x})_i = \mathrm{sign}(x_i)\max(|x_i| - \lambda, 0)$$

Even though the above property is given for real vectors, the same expression generalises naturally to matrices and tensors.

Low-rank subspace

The sub-problem for $\mathbf{T}_n$ is smooth, so we solve it by differentiating the Augmented Lagrangian. After some algebra on $\frac{\partial L}{\partial \mathbf{T}_n}$ we get the following equation for the next iteration of $\mathbf{T}_n$:

$$\alpha_t\mathbf{T}_n + \mu\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r} = \mathbf{U_c}^\mathsf{T}(\mathbf{\Lambda}_n + \mu(\mathbf{X}_n - \mathbf{E}_n))\mathbf{U_r} \tag{4.3}$$

Proof. This is straightforward matrix calculus using either the chain rule or by expanding the inner products, using $\langle \mathbf{X},\, \mathbf{Y} \rangle = \operatorname{tr}\left[\mathbf{X}^\mathsf{T}\mathbf{Y}\right]$ and the formulas for the derivatives of traces. □

We could solve this equation by vectorising both sides then folding the solution back into a matrix:

$$(\alpha_t \mathbf{I}_{r^2} + \mathbf{U_r}^\mathsf{T}\mathbf{U_r} \otimes \mathbf{U_c}^\mathsf{T}\mathbf{U_c})\,\mathrm{vec}(\mathbf{T}_n) = \mathrm{vec}(\mathbf{U_c}^\mathsf{T}(\mathbf{\Lambda}_n + \mu(\mathbf{X}_n - \mathbf{E}_n))\mathbf{U_r})$$

The naive solution consists in left-multiplying by the pseudo-inverse (denoted by †):

$$\mathrm{vec}(\mathbf{T}_n)^* = (\alpha_t \mathbf{I}_{r^2} + \mathbf{U_r}^\mathsf{T}\mathbf{U_r} \otimes \mathbf{U_c}^\mathsf{T}\mathbf{U_c})^\dagger \mathrm{vec}(\mathbf{U_c}^\mathsf{T}(\mathbf{\Lambda}_n + \mu(\mathbf{X}_n - \mathbf{E}_n))\mathbf{U_r})$$

A better, numerically more stable, solution to this kind of problems involves taking a well-chosen matrix decomposition, such as the *QR* or *Cholesky* decomposition, of $(\alpha_t \mathbf{I}_{r^2} + \mathbf{U_r}^\mathsf{T}\mathbf{U_r} \otimes \mathbf{U_c}^\mathsf{T}\mathbf{U_c})$ and solving a simpler linear system. The Matlab `backslash` $\backslash$ operator implements this natively. However, this approach still suffers from a major drawback. The original equation that involved operations on the $r \times r$ square matrix $\mathbf{T}_n$ is transformed into an $r^2 \times r^2$ linear system, on which we perform a decomposition of cubic complexity. In terms of the initial upper bound r on the rank, the time complexity is therefore $O(r^6)$ and the running time quickly becomes prohibitive for moderately large values of r on current hardware.

A much better approach is to recognise that equation (4.3) is a *Discrete-time Generalized Silverster Equation*, and, more precisely, a *Stein* equation of the form:

$$\mathbf{AXB} - \mathbf{X} + \mathbf{Q} = \mathbf{0} \tag{4.4}$$

Several efficient numerical algorithms [54] exist to solve the equation *directly* - that is, without the intermediate transformation into a larger linear system of equations - that are of cubic complexity and avoid the pitfall of having a potentially ill-conditioned matrix after taking the Kronecker product. We can cite the *Bartels-Stewart (B-S)* (Bartels and Stewart [9]) method and the *Hessenberg-Schur (H-S)* (Golub et al. [41]) algorithms; more recent work by Sorensen and Zhou [90] can also be mentioned, as well as algorithms dedicated to particular use cases such as block matrix equations [79], or large equations [16, 62]. We refer to Simoncini [89] for details on the mathematical treatment of Sylvester equations and related.

Our implementation uses the fast `dlyap` routine from the Matlab *Control System Toolbox* which implements the *Hessenberg-Schur* method. It has been shown [41, 79] that the H-S method is more efficient than the B-S method; moreover, the Matlab implementation is based on the highly-optimised SLICOT optimisation and control toolbox [12]. This brings down the time complexity to a much more acceptable $O(r^3)$ and also reduces the storage needed as we no longer need to compute the $r^2 \times r^2$ Kronecker product.

We put equation (4.3) in the required form as follows:

$$-\frac{\mu}{\alpha_t}\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r} - \mathbf{T}_n + \frac{1}{\alpha_t}\mathbf{U_c}^\mathsf{T}(\mathbf{\Lambda}_n + \mu(\mathbf{X}_n - \mathbf{E}_n))\mathbf{U_r} = 0$$

And directly identify each of the components of (4.4).

Left and right bases

The sub-problems associated with $\mathbf{U_c}$ and $\mathbf{U_r}$ are also smooth, so we look for the critical points of the Augmented Lagrangian. We describe the derivations for $\mathbf{U_c}$ and directly give the solution for $\mathbf{U_r}$ as the proof is nearly identical.

The partial derivatives of the relevant components of the Augmented Lagrangian with respect to $\mathbf{U_c}$ are:

$$\frac{\partial}{\partial \mathbf{U_c}}||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_F^2 = -2\alpha_c\mathbf{X}_n\mathbf{U_r}\mathbf{T}_n^\mathsf{T} + 2\mathbf{E}_n\mathbf{U_r}\mathbf{T}_n^\mathsf{T} + 2\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T} \tag{4.5}$$

$$\frac{\partial}{\partial \mathbf{U_c}}\langle \mathbf{\Lambda}_n, \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n\rangle = -\mathbf{\Lambda}_n\mathbf{U_r}\mathbf{T}_n^\mathsf{T} \tag{4.6}$$

It then follows that the optimal $\mathbf{U_c}$ satisfies:

$$\mathbf{U_c}(\alpha_c\mathbf{I} + \mu\sum_n \mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}) = \sum_n(\mu\mathbf{X}_n + \mathbf{\Lambda}_n - \mu\mathbf{E}_n)\mathbf{U_r}\mathbf{T}_n^\mathsf{T} \tag{4.7}$$

Similarly the optimal $\mathbf{U_r}$ satisfies:

$$\mathbf{U_r}(\alpha_r\mathbf{I} + \mu\sum_n \mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{T}_n) = \sum_n(\mu\mathbf{X}_n + \mathbf{\Lambda}_n - \mu\mathbf{E}_n)^\mathsf{T}\mathbf{U_c}\mathbf{T}_n$$

Both systems can be solved in a numerically stable way with the `forward slash /` operator. We could also note that equations (4.5) and (4.7) are *Sylvester equations* $\mathbf{AX} + \mathbf{XB} = \mathbf{C}$.

Updating μ and $\mathbf{\Lambda}$

At each iteration we have:

$$\begin{aligned}\forall n, \quad \mathbf{\Lambda}_n^{k+1} &= \mathbf{\Lambda}_n^k + \mu^k(\mathbf{X}_n^{k+1} - \mathbf{U_c}^{k+1}\mathbf{T}_n^{k+1}\mathbf{U_r}^{k+1} - \mathbf{E}_n^{k+1}) \\ \mu^{k+1} &= \rho\mu^k\end{aligned}$$

We obtain Algorithm 1.

Algorithm 1 Non-orthogonal 2D RPCA

procedure NO2DRPCA($\boldsymbol{\mathcal{X}}, r, \lambda, \alpha_t, \alpha_c, \alpha_r$)
  $\mathbf{U_c}^0, \mathbf{U_r}^0, \boldsymbol{\mathcal{E}}^0, \boldsymbol{\mathcal{T}}^0 \leftarrow$ INITIALIZE($\boldsymbol{\mathcal{X}}$)
  while not converged **do**
    $\boldsymbol{\mathcal{E}}^{t+1} \leftarrow \mathcal{S}_{\lambda/\mu^t}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{T}}^t \times_1 \mathbf{U}_c^t \times_2 \mathbf{U}_r^t + \frac{1}{\mu^t}\mathbf{\Lambda}^t)$
    $\tilde{\boldsymbol{\mathcal{X}}}^{t+1} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^{t+1}$
    $\mathbf{U}_c^{t+1} \leftarrow (\sum_n(\mu^t\tilde{\mathbf{X}}_n^{t+1} + \mathbf{\Lambda}_n^t)\mathbf{U}_r^t(\mathbf{T}_n^t)^\mathsf{T}) \,/\, (\alpha_c\mathbf{I} + \mu^t\sum_n \mathbf{T}_n^t(\mathbf{U}_r^t)^\mathsf{T}\mathbf{U}_r^t(\mathbf{T}_n^t)^\mathsf{T})$
    $\mathbf{U}_r^{t+1} \leftarrow (\sum_n(\mu^t\tilde{\mathbf{X}}_n^{t+1} + \mathbf{\Lambda}_n^t)^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{T}_n^t) \,/\, (\alpha_r\mathbf{I} + \mu^t\sum_n(\mathbf{T}_n^t)^\mathsf{T}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{T}_n^t)$
    for all n **do**
      $\mathbf{T}_n^{t+1} \leftarrow$ STEIN($-\frac{\mu^t}{\alpha_t}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1},\ (\mathbf{U}_r^{t+1})^\mathsf{T}\mathbf{U}_r^{t+1},\ \frac{1}{\alpha_t}(\mathbf{U}_c^{t+1})^\mathsf{T}(\mathbf{\Lambda}_n^t + \mu^t\tilde{\mathbf{X}}_n^{t+1})\mathbf{U}_r^{t+1}$)
    end for
    $\mathbf{\Lambda}^{t+1} \leftarrow \mathbf{\Lambda}^t + \mu^t(\tilde{\boldsymbol{\mathcal{X}}}^{t+1} - \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1})$
    $\mu^{t+1} \leftarrow \rho\mu^t$ ▷ ρ is a parameter that can be tuned
  end while
  return $\mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}}$ ▷ The low-rank tensor is $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1}$
end procedure

4.4 Sparse Non-Orthogonal Robust 2D PCA

In this section we propose a variant of the previous algorithm; the difference is that we chose to further promote sparsity in the low-rank component by setting an element-wise ℓ_1-norm penalty on each frontal slice of $\boldsymbol{\mathcal{T}}$.

4.4.1 Optimisation problem and solution by splitting

We obtain the following non-linear, non-smooth and jointly non-convex optimisation problem:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\left(\alpha_c||\mathbf{U_c}||_{\mathrm{F}}^2 + \alpha_r||\mathbf{U_r}||_{\mathrm{F}}^2\right) + \alpha_t \textstyle\sum_n ||\mathbf{T}_n||_1 + \lambda \sum_n ||\mathbf{E}_n||_1 \\ \text{s.t} \quad & \forall n,\ \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^{\mathsf{T}} + \mathbf{E}_n \end{aligned} \tag{4.8}$$

This problem is difficult to solve as-is since when formulating the Augmented Lagrangian function, we would not be able to rearrange the terms in the objective to obtain an exact proximal step for the non-smooth ℓ_1-norm penalty. Instead of resorting to a linear approximation as described in section 4.5.2, we directly embrace an operator splitting approach and introduce a tensor $\boldsymbol{\mathcal{K}}$ such that, denoting the frontal slices by $\mathbf{K}_n$, problem (4.8) becomes:

$$\begin{aligned} \min \quad & \tfrac{1}{2}\left(\alpha_c||\mathbf{U_c}||_{\mathrm{F}}^2 + \alpha_r||\mathbf{U_r}||_{\mathrm{F}}^2\right) + \alpha_t \textstyle\sum_n ||\mathbf{T}_n||_1 + \lambda \sum_n ||\mathbf{E}_n||_1 \\ \text{s.t} \quad & \forall n,\ \mathbf{X}_n = \mathbf{U_c}\mathbf{K}_n{\mathbf{U_r}}^{\mathsf{T}} + \mathbf{E}_n \\ & \forall n,\ \mathbf{T}_n = \mathbf{K}_n \end{aligned} \tag{4.9}$$

The Augmented Lagrangian of (4.9) is:

$$\begin{aligned} L(\mathbf{U_c}, \mathbf{U_r}, \mathbf{T}, \mathbf{E}, \lambda, \mu, \boldsymbol{\Lambda}) = {} & \frac{\alpha_c}{2}||\mathbf{U_c}||_{\mathrm{F}}^2 + \frac{\alpha_r}{2}||\mathbf{U_r}||_{\mathrm{F}}^2 + \frac{\alpha_t}{2}\sum_n ||\mathbf{T}_n||_{\mathrm{F}}^2 + \lambda \sum_n ||\mathbf{E}_n||_1 + \\ & \sum_n \langle \boldsymbol{\Lambda}_n, \mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^{\mathsf{T}} - \mathbf{E}_n\rangle + \frac{1}{2}\mu||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^{\mathsf{T}} - \mathbf{E}_n||_{\mathrm{F}}^2 + \\ & \sum_n \langle \mathbf{Y}_n, \mathbf{T}_n - \mathbf{K}_n\rangle + \frac{\mu_{\boldsymbol{\mathcal{T}}}}{2}\sum_n ||\mathbf{T}_n - \mathbf{K}_n||_{\mathrm{F}}^2 \end{aligned}$$

Where we introduced a tensor of Lagrange multipliers $\boldsymbol{\mathcal{Y}}$ such that each frontal slice is the matrix of Lagrange multipliers associated with the constraint $\mathbf{T}_n = \mathbf{K}_n$, and a parameter $\mu_{\boldsymbol{\mathcal{T}}}$.

4.4.2 Update equations and algorithm

We derive the updates for $\mathbf{T}_n$ and $\mathbf{K}_n$. The other equations are the same except that $\mathbf{K}_n$ replaces $\mathbf{T}_n$ in all the terms coming from the equality constraints.

Updating the core tensor

By keeping only the terms involving $\mathbf{T}_n$ in the Augmented Lagrangian and rearranging the linear term within the squared Frobenius norm, we find that the optimal $\mathbf{T}_n$ satisfies:

$$\mathbf{T}_n^* = \underset{\mathbf{T}_n}{\operatorname{argmin}} \frac{\alpha_t}{\mu_{\boldsymbol{\mathcal{T}}}}||\mathbf{T}_n||_1 + ||\mathbf{T}_n - \mathbf{K}_n + \frac{1}{\mu_{\boldsymbol{\mathcal{T}}}}\mathbf{Y}_n||_{\mathrm{F}}^2$$

We identify the expression of the proximal operator of the element-wise ℓ_1 norm so:

$$\mathbf{T}_n^* = \mathcal{S}_{\alpha_t/\mu_{\boldsymbol{\mathcal{T}}}}(\mathbf{K}_n - \frac{1}{\mu_{\boldsymbol{\mathcal{T}}}}\mathbf{Y}_n)$$

Updating $\mathbf{K}_n$

The optimal $\mathbf{K}_n$ is solution to a least-squares problem similar to that of $\mathbf{T}_n$ in Algorithm 1, with the addition of $\mathbf{T}_n$ and its Lagrange multiplier $\mathbf{Y}_n$:

$$\mu_{\boldsymbol{\mathcal{T}}}\mathbf{K}_n + \mu\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{K}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r} = \mathbf{U_c}^\mathsf{T}(\mathbf{\Lambda}_n + \mu(\mathbf{X}_n - \mathbf{E}_n))\mathbf{U_r} + \mu_{\boldsymbol{\mathcal{T}}}\mathbf{T}_n + \mathbf{Y}_n \tag{4.10}$$

This is again a Stein equation that can be solved as previously explained:

$$-\frac{\mu}{\mu_{\boldsymbol{\mathcal{T}}}}\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{K}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r} - \mathbf{K}_n + \frac{1}{\mu_{\boldsymbol{\mathcal{T}}}}\left(\mathbf{U_c}^\mathsf{T}(\mathbf{\Lambda}_n + \mu(\mathbf{X}_n - \mathbf{E}_n))\mathbf{U_r} + \mathbf{Y}_n\right) + \mathbf{T}_n = 0 \tag{4.11}$$

We obtain Algorithm 2

Algorithm 2 Sparse Non-orthogonal 2D RPCA

procedure SPARSE_NO2DRPCA($\boldsymbol{\mathcal{X}}, r, \lambda, \alpha_t, \alpha_c, \alpha_r$)
  $\mathbf{U_c}^0, \mathbf{U_r}^0, \boldsymbol{\mathcal{E}}^0, \boldsymbol{\mathcal{T}}^0, \boldsymbol{\mathcal{K}}^0 \leftarrow$ INITIALIZE($\boldsymbol{\mathcal{X}}$)
  while not converged **do**
    $\boldsymbol{\mathcal{E}}^{t+1} \leftarrow \mathcal{S}_{\lambda/\mu^t}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{K}}^t \times_1 \mathbf{U}_c^t \times_2 \mathbf{U}_r^t + \frac{1}{\mu^t}\mathbf{\Lambda}^t)$
    $\tilde{\boldsymbol{\mathcal{X}}}^{t+1} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^{t+1}$
    $\mathbf{U}_c^{t+1} \leftarrow (\sum_n(\mu^t\tilde{\mathbf{X}}_n^{t+1} + \mathbf{\Lambda}_n^t)\mathbf{U}_r^t(\mathbf{K}_n^t)^\mathsf{T}) \,/\, (\alpha_c\mathbf{I} + \mu^t\sum_n \mathbf{K}_n^t(\mathbf{U}_r^t)^\mathsf{T}\mathbf{U}_r^t(\mathbf{K}_n^t)^\mathsf{T})$
    $\mathbf{U}_r^{t+1} \leftarrow (\sum_n(\mu^t\tilde{\mathbf{X}}_n^{t+1} + \mathbf{\Lambda}_n^t)^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{K}_n^t) \,/\, (\alpha_r\mathbf{I} + \mu^t\sum_n(\mathbf{K}_n^t)^\mathsf{T}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{K}_n^t)$
    for all n **do**
      $\mathbf{K}_n^{t+1} \leftarrow$ STEIN($-\frac{\mu^t}{\mu_{\boldsymbol{\mathcal{T}}}^t}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}$, $(\mathbf{U}_r^{t+1})^\mathsf{T}\mathbf{U}_r^{t+1}$, $\frac{1}{\mu_{\boldsymbol{\mathcal{T}}}^t}\left[(\mathbf{U}_c^{t+1})^\mathsf{T}(\mathbf{\Lambda}_n^t + \mu^t\tilde{\mathbf{X}}_n^{t+1})\mathbf{U}_r^{t+1} + \mathbf{Y}_n^t\right] + \mathbf{T}_n^t$)
      $\mathbf{T}_n^{t+1} \leftarrow \mathcal{S}_{\alpha_t/\mu_{\boldsymbol{\mathcal{T}}}^t}(\mathbf{K}_n^{t+1} - \frac{1}{\mu_{\boldsymbol{\mathcal{T}}}^t}\mathbf{Y}_n^t)$
    end for
    $\mathbf{\Lambda}^{t+1} \leftarrow \mathbf{\Lambda}^t + \mu^t(\tilde{\boldsymbol{\mathcal{X}}}^{t+1} - \boldsymbol{\mathcal{K}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1})$
    $\boldsymbol{\mathcal{Y}}^{t+1} \leftarrow \boldsymbol{\mathcal{Y}}^t + \mu_{\boldsymbol{\mathcal{T}}}^t(\boldsymbol{\mathcal{T}}^{t+1} - \boldsymbol{\mathcal{K}}^{t+1})$
    $\mu^{t+1} \leftarrow \rho\mu^t$ ▷ ρ is a parameter that can be tuned
    $\mu_{\boldsymbol{\mathcal{T}}}^{t+1} \leftarrow \rho\mu_{\boldsymbol{\mathcal{T}}}^t$
  end while
  return $\mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}}$ ▷ The low-rank tensor is $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1}$
end procedure

4.5 Group Lasso-penalised bases

In the previous sections, we put Tikhonov regularisation on the matrices of the left and right bases $\mathbf{U_c}$ and $\mathbf{U_r}$. The rationale is that, the Frobenius norm being the same as the **Schatten-2** norm, we penalise the magnitude of the singular values and therefore promote a low-rank.

In this section, we study an alternative way of reducing the dimensionality of $\mathrm{Span}(\mathbf{U_c})$ and $\mathrm{Span}(\mathbf{U_r})$ and chose to promote *group sparsity* in $\mathbf{U_c}$ and $\mathbf{U_r}$ by putting a Group Lasso penalty, where each column is a group of variables. This approach is fundamentally different since it is based on the structure of the matrix and not on its spectrum; moreover, there is no guarantee in general that the non-zero columns will all be linearly independent, so it is theoretically possible that the rank of $\mathbf{U_c}$ or $\mathbf{U_r}$ be **lower** than their number of non-zero columns. Element-wise sparsity alone would not make sense since it would drive elements to zero individually, while our goal is to prune irrelevant dimensions by driving whole columns to zero at once.

4.5.1 Optimisation problem

Let us first recall the expression of the non-overlapping group lasso penalty, or ℓ_1/ℓ_2 norm.

Definition 4.5.1. Let $\boldsymbol{\mathcal{A}} = \{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ a set of n disjoint groups of variables. The group lasso on $\boldsymbol{\mathcal{A}}$ is:

$$||\boldsymbol{\mathcal{A}}||_{\ell_1/\ell_2} = \sum_{i=1}^{n} ||\mathbf{a}_i||_2$$

Which is the ℓ_1 norm of the vector of the ℓ_2 norms of the groups.

We solve the following non-linear, non-smooth and jointly non-convex constrained problem:

$$\min \alpha_c||\mathbf{U_c}||_{\ell_1/\ell_2} + \alpha_r||\mathbf{U_r}||_{\ell_1/\ell_2} + \lambda \textstyle\sum_n ||\mathbf{E}_n||_1$$
$$\text{s.t } \forall n,\ \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} + \mathbf{E}_n$$

Again, we choose to approach the problem by convex optimisation in the ADMM framework. We first describe a modification of the ADMM algorithm, called *linearised* ADMM, to avoid introducing additional variables. This approach is very sensitive to numerical errors and introduces more additional complexity than splitting, as will be explained in the below section. Therefore, we derive the splitting updates and finally propose an ADMM algorithm for the problem.

4.5.2 Solution with an inexact proximal gradient step

We describe an inexact proximal gradient step that doesn't require the addition of new constraints. The approach is based on linearisation of the sub-problem and is comparable to *proximal gradient* methods such as FISTA (Beck and Teboulle [11]). We describe the approach for $\mathbf{U_c}$ only as it directly translates to $\mathbf{U_r}$.

Similarly to how we solved for $\mathbf{E}_n$, we can show that:

$$\underset{\mathbf{U_c}}{\operatorname{argmin}}\, L(\dots) = \underset{\mathbf{U_c}}{\operatorname{argmin}}\, ||\mathbf{U_c}||_{\ell_1/\ell_2} + \frac{\alpha_c}{\mu}\sum_n ||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n + \frac{1}{\mu}\mathbf{\Lambda}_n||_\mathrm{F}^2$$

Which is **not** the expression of a proximal operator because of the sum of quadratic penalties on functions of $\mathbf{U_c}$.

Let $f(\mathbf{U_c}) = \frac{\alpha_c}{\mu}\sum_n ||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2$, then a linearised proximal update for $\mathbf{U_c}$ at step t is given by

$$\mathbf{U}_r^{t+1} = \operatorname{prox}_{\alpha_c/\mu||.||_{\ell_1/\ell_2}}(\mathbf{U}_r^t - \frac{1}{\mu L^t}\nabla f({\mathbf{U_c}}^t))$$

Where L^t needs to be larger than the Lipschitz constant of f. In this case, we may choose:

$$L^t > |||\nabla^2 f(\mathbf{U_c})|||$$

But a better way is to find a suitable L^t at each iteration with a line search strategy. Bach et al. [7] give a modified Armijo condition that can be used with backtracking.

4.5.3 Solution using ADMM and splitting

The choice of the largest singular value of the Hessian for L^t is ad-hoc and is guaranteed to be correct only when $\nabla^2 f(\mathbf{U_c}) = \alpha\mathbf{I}$. Instead, we can use a standard ADMM solution by introducing $\mathbf{A}, \mathbf{B}$ and forcing $\mathbf{U_c} = \mathbf{A}, \mathbf{U_r} = \mathbf{B}$.

Formulating the Augmented Lagrangian and using the same technique of merging the linear and quadratic terms, we find that the newly introduced constraints, $\mathbf{U_c} = \mathbf{A}$ and $\mathbf{U_r} = \mathbf{B}$ induce quadratic terms $||\mathbf{U_c} - \mathbf{A}||_\mathrm{F}$ and $||\mathbf{U_r} - \mathbf{B}||_\mathrm{F}$ and hence yield exact proximal steps.

Solution for $\mathbf{U_c}$ and $\mathbf{U_r}$

The optimal $\mathbf{U_c}$ is solution of the below optimisation problem:

$$\mathbf{U_c}^* = \underset{\mathbf{U_c}}{\operatorname{argmin}} \frac{\alpha_c}{\mu_c}||\mathbf{U_c}||_{\ell_1/\ell_2} + ||\mathbf{U_c} - \mathbf{A} + \frac{1}{\mu_c}\mathbf{\Lambda}_c||_{\mathrm{F}}^2 \tag{4.12}$$

We identify the expression of the proximal operator of the structured ℓ_1/ℓ_2 norm, whose closed form expression is given by Proposition 4.5.1.

Proposition 4.5.1. *The scaled proximal operator of the ℓ_1/ℓ_2 is the function:*

$$\operatorname{prox}_{\lambda||.||_{\ell_1/\ell_2}}(\mathbf{x}) = \underset{\mathbf{v}}{\operatorname{argmin}}\, ||\mathbf{v}||_{\ell_1/\ell_2} + ||\mathbf{x} - \mathbf{v}||_{\mathrm{F}}^2$$

And has the following closed-form expression:

$$\operatorname{prox}_{\lambda||.||_{\ell_1/\ell_2}}(\mathbf{x}) = \begin{cases} \mathbf{x} - \lambda\frac{\mathbf{x}}{||\mathbf{x}||_2}, & ||\mathbf{x}||_2 > \lambda \\ 0, & ||\mathbf{x}||_2 \leq \lambda \end{cases}$$

So the optimal $\mathbf{U_c}$ is:

$$\mathbf{U_c}^* = \operatorname{prox}_{\lambda||.||_{\ell_1/\ell_2}}(\mathbf{A} - \frac{1}{\mu_c}\mathbf{\Lambda}_c)$$

Similarly to $\mathbf{U_c}$ we get:

$$\mathbf{U_r}^* = \operatorname{prox}_{\lambda||.||_{\ell_1/\ell_2}}(\mathbf{B} - \frac{1}{\mu_r}\mathbf{\Lambda}_r)$$

Update equations for $\mathbf{A}$ and $\mathbf{B}$

In this subsection, $\mathcal{T}$ refers to either the actual core tensor $\mathcal{T}$ as in section 1 or to the split variable $\mathcal{K}$ if we choose an ℓ_1 penalty on the core such as in section 4.4.

The updates for the split variables $\mathbf{A}$ and $\mathbf{B}$ are very similar to the original updates of the bases, with the addition of $\mathbf{U_c}$ or $\mathbf{U_r}$ and of the Lagrange multipliers $\mathbf{\Lambda}_c$ or $\mathbf{\Lambda}_r$. After some algebra on the partial derivative of the Augmented Lagrangian with respect to $\mathbf{A}$ we get:

$$\mathbf{A}(\mu_c\mathbf{I} + \mu\sum_n \mathbf{T}_n\mathbf{B}^\mathsf{T}\mathbf{B}\mathbf{T}_n^\mathsf{T}) = \mu_c\mathbf{U_c} + \mathbf{\Lambda}_c + \mu\sum_n(\mathbf{X}_n + \frac{1}{\mu}\mathbf{\Lambda}_n - \mathbf{E}_n)\mathbf{B}\mathbf{T}_n^\mathsf{T} \tag{4.13}$$

Similarly, we find that the optimal update for $\mathbf{B}$ is solution of:

$$\mathbf{B}(\mu_r\mathbf{I} + \mu\sum_n \mathbf{T}_n^\mathsf{T}\mathbf{A}^\mathsf{T}\mathbf{A}\mathbf{T}_n) = \mu_r\mathbf{U_r} + \mathbf{\Lambda}_r + \mu\sum_n(\mathbf{X}_n + \frac{1}{\mu}\mathbf{\Lambda}_n - \mathbf{E}_n)^\mathsf{T}\mathbf{A}\mathbf{T}_n \tag{4.14}$$

These linear equations can be solved efficiently using a variety of solvers depending on the numerical properties of the matrices. We obtain Algorithm 3.

4.6 Iterative robust estimation of the mean

A usual preprocessing step for non-robust dimensionality reduction methods such as PCA involves subtracting the mean of the data. Working on zero-mean data improves the performance of these algorithms by avoiding numerical underflow.

Algorithm 3 Non-orthogonal 2D RPCA with Group Lasso

procedure NO2DRPCA_GL($\boldsymbol{\mathcal{X}}, r, \lambda, \alpha_t, \alpha_c, \alpha_r$)
  $\mathbf{U_c}^0, \mathbf{U_r}^0, \boldsymbol{\mathcal{E}}^0, \boldsymbol{\mathcal{T}}^0 \leftarrow$ INITIALIZE($\boldsymbol{\mathcal{X}}$)
  $\mathbf{A}^0, \mathbf{B}^0 \leftarrow \mathbf{U_c}^0, \mathbf{U_r}^0$
  while not converged **do**
    $\boldsymbol{\mathcal{E}}^{t+1} \leftarrow \mathcal{S}_{\lambda/\mu^t}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{T}}^t \times_1 \mathbf{A}^t \times_2 \mathbf{B}^t + \frac{1}{\mu^t}\boldsymbol{\Lambda}^t)$
    $\tilde{\boldsymbol{\mathcal{X}}}^{t+1} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^{t+1}$
    $\mathbf{A}^{t+1} \leftarrow (\sum_n(\mu^t\tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t)\mathbf{B}^t(\mathbf{T}_n^t)^\mathsf{T} + \mu_c^t\mathbf{U}_c^t + \boldsymbol{\Lambda}_c^t) \,/\, (\mu_c^t\mathbf{I} + \mu^t\sum_n \mathbf{T}_n^t(\mathbf{B}^t)^\mathsf{T}\mathbf{B}^t(\mathbf{T}_n^t)^\mathsf{T})$
    $\mathbf{B}^{t+1} \leftarrow (\sum_n(\mu^t\tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t)^\mathsf{T}\mathbf{A}^{t+1}\mathbf{T}_n^t + \mu_r^t\mathbf{U}_r^t + \boldsymbol{\Lambda}_r^t) \,/\, (\mu_r^t\mathbf{I} + \mu^t\sum_n(\mathbf{T}_n^t)^\mathsf{T}(\mathbf{A}^{t+1})^\mathsf{T}\mathbf{A}^{t+1}\mathbf{T}_n^t)$
    $\mathbf{U}_c^{t+1} \leftarrow \mathrm{prox}_{\lambda\|.\|_{\ell_1/\ell_2}}(\mathbf{A}^{t+1} - \frac{1}{\mu_c^t}\boldsymbol{\Lambda}_c^t)$
    $\mathbf{U}_r^{t+1} \leftarrow \mathrm{prox}_{\lambda\|.\|_{\ell_1/\ell_2}}(\mathbf{B}^{t+1} - \frac{1}{\mu_r^t}\boldsymbol{\Lambda}_r^t)$
    for all n **do**
      $\mathbf{T}_n^{t+1} \leftarrow$ UPDATE_T_STEIN(...)
    end for
    $\boldsymbol{\Lambda}^{t+1} \leftarrow \boldsymbol{\Lambda}^t + \mu^t(\tilde{\boldsymbol{\mathcal{X}}}^{t+1} - \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1})$
    $\boldsymbol{\Lambda}_c^{t+1} \leftarrow \boldsymbol{\Lambda}_c^t + \mu_c^t(\mathbf{U}_c^{t+1} - \mathbf{A}^{t+1})$
    $\boldsymbol{\Lambda}_r^{t+1} \leftarrow \boldsymbol{\Lambda}_r^t + \mu_r^t(\mathbf{U}_r^{t+1} - \mathbf{B}^{t+1})$
    $\mu^{t+1} \leftarrow \rho\mu^t$ ▷ ρ is a parameter that can be tuned
    $\mu_c^{t+1} \leftarrow \rho\mu_c^t$
    $\mu_r^{t+1} \leftarrow \rho\mu_r^t$
  end while
  return $\mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}}$ ▷ The low-rank tensor is $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1}$
end procedure

The empirical mean has good properties as a non-parametric estimator of the true mean of an unknown distribution. It is an unbiased estimator and converges to the true mean, both almost surely and in mean-square. Moreover, the empirical mean is also the maximum likelihood estimator of the mean for some of the most usual and useful distributions, such as the ubiquitous Normal distribution. Maximum likelihood estimators are important because they are **strongly convergent** estimators, are **asymptotically unbiased, Gaussian, and efficient**. For an estimator to be (asymptotically) efficient means that the variance of the estimator is (asymptotically) equal to the **Cramer-Rao lower bound**, *i.e,* that there exist no estimator with lower variance. It is also easy to show that in the case of Normally distributed data - a very common assumption - the sample mean is an **unbiased and efficient estimator of the mean** of the Normal distribution. Finally, the asymptotic normality of maximum likelihood estimators allows them to be used in conjunction of the **delta method** for estimating functions of the unknown estimated quantity under very mild assumptions.

Unfortunately, in the presence of noise and outliers, it is no longer possible to accurately estimate the true mean by the sample mean and a **robust estimator** of the mean is needed. A simple method is to use the sample **median** instead of the mean as a measure of centrality, but is still quite primitive and has been unsatisfactory in practice in our case. More refined methods for robust estimation include trimmed means, which automatically discard a certain percentage of the observations, L-estimators, and M-estimators. L-estimators generally provide a good trade-off between efficiency and computational complexity while modern M-estimators are more efficient and generally preferred when possible, although they are much more difficult to compute. M-estimators for Robust PCA on both matrices and tensors have been studied by Yang et al. [110].

An alternative method that proves useful in our case is to remove the outliers from the sample

before computing the standard empirical mean. Classically, statisticians have done so by manual inspection of the sample, which is obviously not an option for automated processing. However, one of the very goals of this study *is* to algorithmically separate the outliers from the low-rank component. This leads us to propose a very simple iterative method for robustly estimating the sample mean. At each step, we estimate the mean $\hat{\mathbf{M}}$ of the frontal slices by computing the empirical mean along the third dimension of the observations *minus* the estimated outliers, as formalised by equation 4.15.

$$\hat{\mathbf{M}}^t = \frac{1}{N}\sum_n (\mathbf{X}_n - \mathbf{E}_n^t) \tag{4.15}$$

Where N denotes the sample size. This method has been successfully used in Mateos and Giannakis [69], and requires very little modifications of our algorithms. In fact, besides the addition of the mean-estimation step, *all the subsequent steps remain the same* and only the temporary data are changed from $\tilde{\boldsymbol{\mathcal{X}}}^t = \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^t$ to $\tilde{\boldsymbol{\mathcal{X}}}^t = \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^t - \mathbf{M}^t$. We experimentally demonstrate that this additional step improves the performance of the algorithm on background modelling and image reconstruction tasks.

We give a modified version of Algorithm 1 with the robust estimation of the mean included as Algorithm 4.

Algorithm 4 Non-orthogonal 2D RPCA with robust estimation of the mean

procedure NO2DRPCA_WITH_MEAN($\boldsymbol{\mathcal{X}}, r, \lambda, \alpha_t, \alpha_c, \alpha_r$)
  $\mathbf{U_c}^0, \mathbf{U_r}^0, \boldsymbol{\mathcal{E}}^0, \boldsymbol{\mathcal{T}}^0 \leftarrow$ INITIALIZE($\boldsymbol{\mathcal{X}}$)
  $\mathbf{M}^0 \leftarrow \mathbf{0}$
  while not converged **do**
    $\boldsymbol{\mathcal{E}}^{t+1} \leftarrow \mathcal{S}_{\lambda/\mu^t}(\boldsymbol{\mathcal{X}} - \mathbf{M}^t - \boldsymbol{\mathcal{T}}^t \times_1 \mathbf{U}_c^t \times_2 \mathbf{U}_r^t + \frac{1}{\mu^t}\boldsymbol{\Lambda}^t)$
    $\mathbf{M}^{t+1} \leftarrow \frac{1}{N}\sum_n (\mathbf{X}_n - \mathbf{E}_n^{t+1})$
    $\tilde{\boldsymbol{\mathcal{X}}}^{t+1} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^{t+1} - \mathbf{M}^{t+1}$
    $\mathbf{U}_c^{t+1} \leftarrow (\sum_n (\mu^t \tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t)\mathbf{U}_r^t(\mathbf{T}_n^t)^\mathsf{T}) \,/\, (\alpha_c \mathbf{I} + \mu^t \sum_n \mathbf{T}_n^t (\mathbf{U}_r^t)^\mathsf{T}\mathbf{U}_r^t(\mathbf{T}_n^t)^\mathsf{T})$
    $\mathbf{U}_r^{t+1} \leftarrow (\sum_n (\mu^t \tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t)^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{T}_n^t) \,/\, (\alpha_r \mathbf{I} + \mu^t \sum_n (\mathbf{T}_n^t)^\mathsf{T}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{T}_n^t)$
    for all n **do**
      $\mathbf{T}_n^{t+1} \leftarrow$ STEIN($-\frac{\mu^t}{\alpha_t}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1},\ (\mathbf{U}_r^{t+1})^\mathsf{T}\mathbf{U}_r^{t+1},\ \frac{1}{\alpha_t}(\mathbf{U}_c^{t+1})^\mathsf{T}(\boldsymbol{\Lambda}_n^t + \mu^t \tilde{\mathbf{X}}_n^{t+1})\mathbf{U}_r^{t+1}$)
    end for
    $\boldsymbol{\Lambda}^{t+1} \leftarrow \boldsymbol{\Lambda}^t + \mu^t(\tilde{\boldsymbol{\mathcal{X}}}^{t+1} - \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1})$
    $\mu^{t+1} \leftarrow \rho\mu^t$ ▷ ρ is a parameter that can be tuned
  end while
  return $\mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}}, \mathbf{M}$ ▷ The low-rank tensor is $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1} + \mathbf{M}$
end procedure

4.7 Model validation on synthetic data

In order to prove the correctness of the concept and implementation of our 2D algorithms, we devised a set of numerical experiments on synthetic data designed to challenge the model. We first describe how we generated the test data, and then provide an assessment of the four methods to prove their ability to recover exactly the low-rank component.

In this section and in future chapters, we will use the abbreviations given in Table 4.1. When robust estimation of the mean is enabled, we will suffix the methods by a capital M, as in *RPCA2D Fro ℓ_2 M*.

Core vs Bases	ℓ_2	ℓ_1/ℓ_2
ℓ_2	RPCA2D Fro ℓ_2	RPCA2D GL ℓ_2
ℓ_1	RPCA2D Fro ℓ_1	RPCA2D GL ℓ_1

Table 4.1: Abbreviations used for our four methods

4.7.1 Experimental setup

Recall that the assumption we made is that a collection of images sampled from the same distribution could be factorised in a special case of *Tucker2* decomposition where we would find optimal bases and a core-tensor to store the specific information for each matrix, with the orthogonality constraints on the bases relaxed. We also assume that the noise component is sparse and Laplacian. In order to test our model's ability to recover data that match these assumptions, we generated synthetic data as follows:

- We sample random matrices $\mathbf{U_c} \in \mathbb{R}^{n \times r_c}$ and $\mathbf{U_r} \in \mathbb{R}^{m \times r_r}$ from a multivariate uniform distribution $\mathcal{U}(0, 1)$, we choose $m = n = 1000$
- We sample a corresponding dense low-dimensional tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{r_c \times r_r \times N}$ where N is the number of observations chosen to be $N = 100$
- The original low-rank component $\boldsymbol{\mathcal{L}}$ is taken as $\boldsymbol{\mathcal{L}} = (\boldsymbol{\mathcal{T}} \times_1 \mathbf{U_c} \times_2 \mathbf{U_r})/\sqrt{r_c r_r}$ to keep the magnitude of the elements low
- We sample additive sparse noise $\boldsymbol{\mathcal{E}}$ at the 30% and 60% levels by, for each matrix, keeping the required percentage of elements from $\text{sign}(\mathbf{E}_0)$ with $\mathbf{E}_0 \sim \mathcal{N}(0, \mathbf{I})$. This gives us sparse noise of magnitude either 1 or -1
- The final noisy observation is $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{E}}$

We then test a range of values for the regularisation weight λ while keeping the other parameters fixed. The precise values of the parameters for each algorithm will be reported in the corresponding sub-section.

Numerical approximation of the rank

Recall the rank of a matrix can be defined equivalently as the dimension of the space spanned by its columns and by the number of non-zero singular values in its SVD. Numerically, however, it is usual that a matrix be *approximately low-rank*. We will use this notion extensively and hereby give a definition of what we mean by an approximately low-rank matrix based on the Schatten-∞ norm.

Definition 4.7.1 (Approximately low-rank matrix at a given threshold)**.** Let $\mathbf{X}$ a matrix and $\mathbf{USV}^\top$ its Singular Value Decomposition. We will say that $\mathbf{X}$ is *approximately of rank r at threshold δ* if and only if

$$\text{card}(\{s \in \mathbf{S} : |s| \geq \delta ||\mathbf{X}||_\infty\}) = r$$

With $||.||_\infty$ the Schatten-∞ norm (largest singular value).

4.7.2 Sample run and automated tuning

We consider the four models where the penalty on the bases is either the ℓ_2 (Frobenius), or ℓ_1/ℓ_2 (Group Lasso) norm and test both the effect of an ℓ_1 and ℓ_2 regularisation on the core tensor $\boldsymbol{\mathcal{T}}$.

Our experiment consists in varying the value of λ while keeping other variables fixed, the choice of parameters is the same for both the ℓ_1 and ℓ_2 cases:

- $\alpha_t = 1e-2$
- $\alpha_c = \alpha_r = 1$ for the Frobenius case
- $\alpha_c = \alpha_r = 1e-3$ for the Group Lasso case
- $\rho = 1.2$
- Convergence threshold: $\delta = 1e-7$
- Robust estimator of the mean: disabled

In both cases, we suppose a **loose** upper bound r on the ranks r_c and r_r to be known. For this experiment, we choose $r_c = 23$, $r_r = 41$ and $r = 100$. Our goal is to keep the computation time reasonable and to model how expert knowledge can help the algorithms perform, while assuming as little prior information as possible.

Overview: test of a simple criterion for choosing λ

As a short prequel, we first test the recovery with $\lambda = \frac{1}{\sqrt{N \max(m,n)}}$ and measure the PSNR, SSIM, FSIM and relative error in ℓ_2-norm for the two methods that use a Frobenius penalty on the bases. The results are presented in table (4.2).

	ℓ_2 (30% noise)	ℓ_1 (30% noise)	ℓ_2 (60% noise)	ℓ_1 (60% noise)
PSNR	153.3009	156.1967	147.4913	150.1217
SSIM	1	1	1	1
FSIM	1	1	1	1
MSAM	2.9197e-08	2.1675e-08	5.9671e-08	6.0703e-08
Relative error	1.6601e-07	1.2952e-07	3.1978e-07	3.3198e-07
Recovered r_c	23	23	23	23
Recovered r_r	41	41	41	41
Number of iterations	56	87	63	85

Table 4.2: Recovery performance of *RPCA2D Fro* ℓ_2 and *RPCA2D Fro* ℓ_1 with $\lambda = \frac{1}{\sqrt{N \max(m,n)}}$ averaged over all 100 matrices

By all metrics, both methods recovered the low-rank component exactly in the 30% and in the 60% noise cases, although the ℓ_2-regularised version did it in fewer iterations than the ℓ_1-regularised version. In the 30% noise case, the ℓ_1-regularised version attained 21.98% lower relative error and 25.76% lower MSAM while having a higher PSNR. In the 60% noise case the ℓ_1 version maintained a higher PSNR, the ℓ_2 algorithm had a 1.7% lower relative error and both MSAM were equal.

Both methods recovered the true ranks at the $1e-3$ threshold level at the two noise levels. Both structural similarity index (SSIM and FSIM) indicated that the 100 matrices were exactly recovered at both noise levels.

Automated tuning of λ

The previous overview tested a specific choice of λ for two out of our four methods and assessed the recovery of the low-rank component only. In this experiment, we automatically try a range of values for λ and report the evolution of several quality metrics for all of our algorithms. The metrics we chose to report are:

- The relative ℓ_2 error and the FSIM for the low rank component
- The relative ℓ_2 error and the density for the sparse component

The FSIM and the density are indicators of the *structure* of the components. The range of values we test for λ is between $2e-4$ and $2e-1$. We conclude by presenting the evolution of the MSAM in both cases for indicative purposes.

Recovery in the 30% noise case We report the evolution of the relative error on the low-rank and sparse components in figure (4.1), and the evolution of the FSIM and density indexes in figure (4.2).

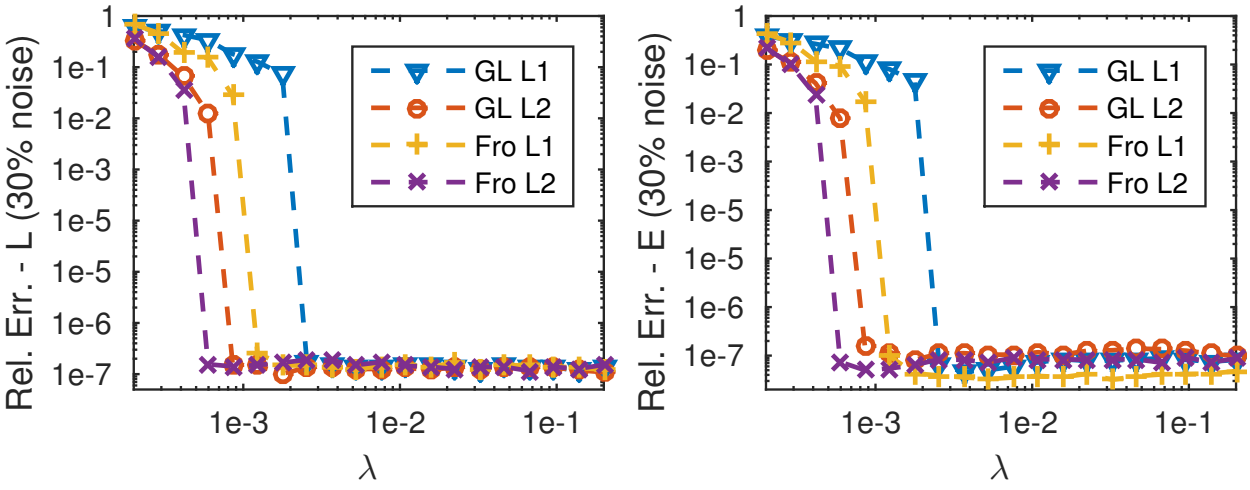

Figure 4.1: Relative reconstruction error of the low-rank (left) and sparse (right) components for 30% noise on synthetic data

The four measures clearly show near-exact recovery of the two components is attained by all algorithms on a wide range of values for λ. Actually, we observe a *phase transition* phenomenon similar to the one described in Goldfarb and Qin [40]: above a certain threshold - that depends on the algorithm - the reconstruction error is nearly zero. All four methods reach an FSIM of 1 on the low-rank component, demonstrating high-quality recovery of the structural information. All methods except *RPCA2D GL* ℓ_2 successfully recover the density of the sparse component and even the **exact number of non-zero entries**. The apparent contradiction between the fact that *RPCA2D GL* ℓ_2 achieves close-to-zero relative reconstruction error on the sparse tensor but doesn't reach its original sparsity level means many small but non-zero elements remain in the reconstructed $\boldsymbol{\mathcal{E}}$.

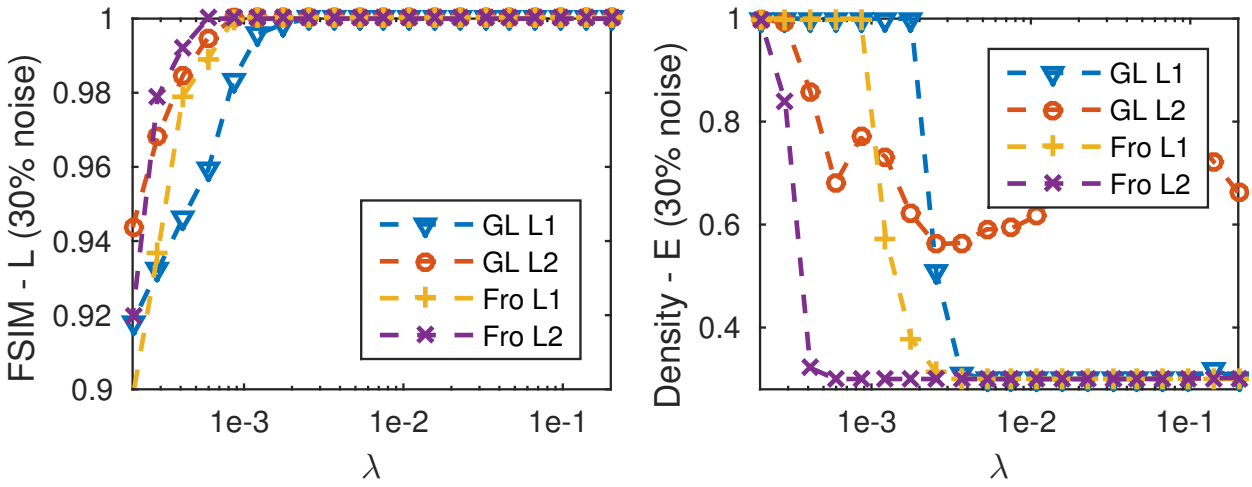

Figure 4.2: Structural recovery of the low-rank (left) and sparse (right) components for 30% noise on synthetic data

Recovery in the 60% **noise case** Similarly to the 30% noise case we plot the metrics in figures (4.3) and (4.4).

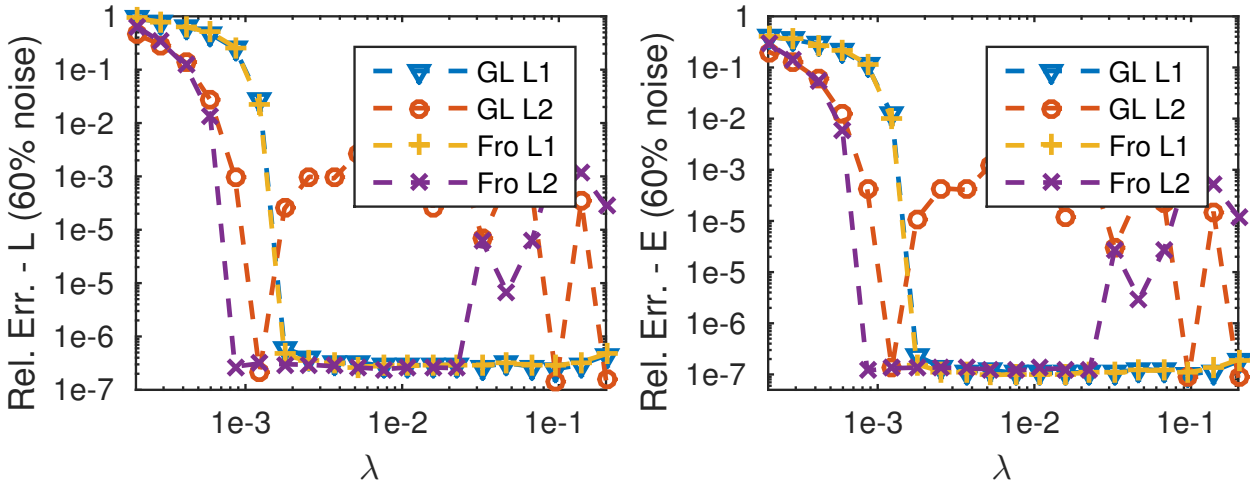

Figure 4.3: Relative reconstruction error of the low-rank (left) and sparse (right) components for 60% noise on synthetic data

Even at this gross-corruption level, all four methods are still able to recover the two components exactly for well-chosen values of λ.

The methods using an ℓ_2 regularisation on $\boldsymbol{\mathcal{T}}$ exhibit an erratic behaviour for both $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{E}}$. *RPCA2D Fro* ℓ_2 reaches a plateau of low reconstruction error for $\lambda \in [8e{-}4, 2e{-}2]$ and then becomes unstable. *RPCA2D GL* ℓ_2 is unstable over the whole range of values we tested, but interestingly attains the lowest relative error of all four methods for three distinct values of λ. On the contrary, the ℓ_1-penalised methods have the same behaviour as in the 30% noise case.

Regarding the structural recovery, the measured mean FSIM index shows the four methods still give high-quality reconstruction of $\boldsymbol{\mathcal{L}}$ for a wide range of values of λ, even when the relative reconstruction error is high for *RPCA2D Fro* ℓ_2 and *RPCA2D GL* ℓ_2. Clearer differences appear for the density of $\boldsymbol{\mathcal{E}}$: the best-performing methods are *RPCA2D Fro* ℓ_1 and *RPCA2D GL* ℓ_1 and are still able to exactly determine the number of non-zero elements of $\boldsymbol{\mathcal{E}}$. As in the 30% noise case, *RPCA2D*

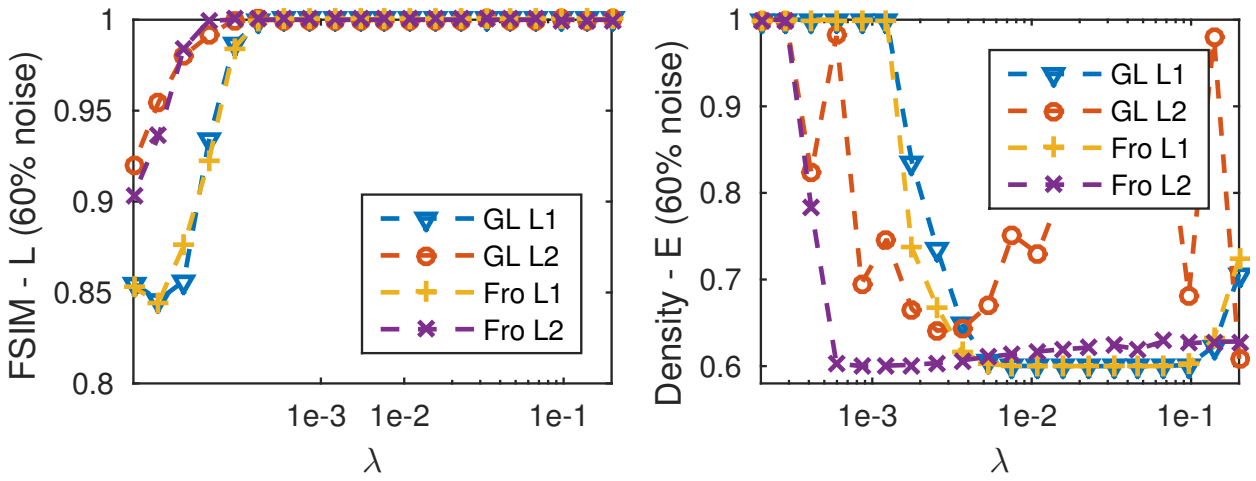

Figure 4.4: Structural recovery of the low-rank (left) and sparse (right) components for 60% noise on synthetic data

GL ℓ_2 doesn't perform well with regards to this measure. *RPCA2D Fro* ℓ_2 closely approximates the true density level, although for fewer values of λ than in the 30% noise case.

Based on these observations, we expect the methods based on ℓ_1-regularisation on $\mathcal{T}$ to be more robust than their ℓ_2 counterparts on real-world data at high level of noise. The comparative study of our method and several others from the sate of the art is the object of Chapter 5 of this thesis.

Quality of the estimation of the mean

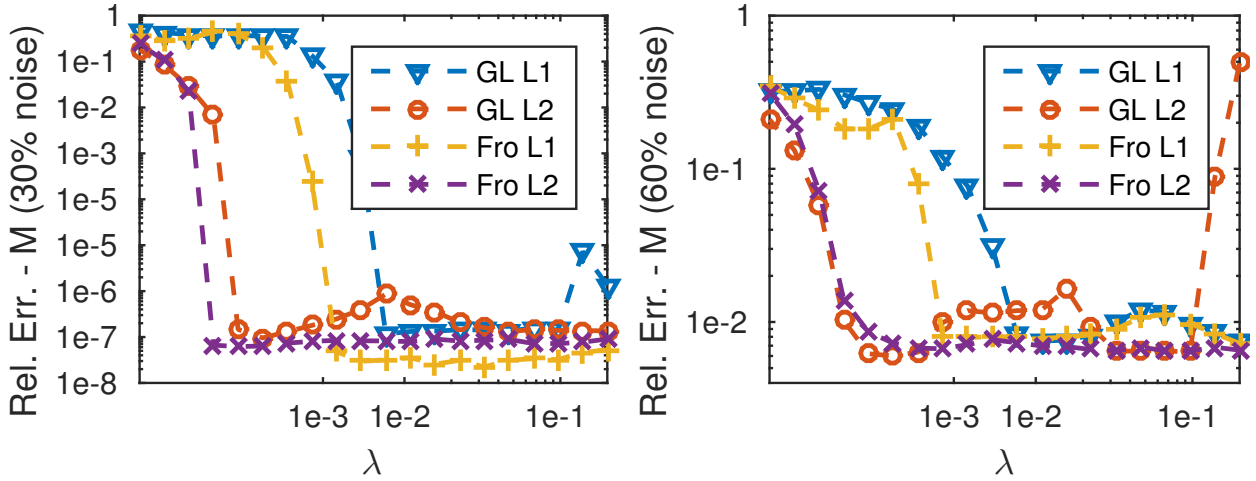

Figure 4.5: Relative error on the sample mean at the 30% and 60% noise levels

In order to validate our procedure for robustly estimating the mean of the data in presence of noise, we report the relative reconstruction error of the mean in the 30% and 60% noise cases.

As shown in figure (4.5) our methods converge to the true mean in the 30% noise case but not in the 60% noise case. This is to be expected and we believe more sophisticated approaches are needed to handle gross corruption. The methods that use the spectral approach to rank-minimisation tend to perform better on this benchmark.

4.7.3 Sparsity of the solution and rank recovery

We verified that all four methods are able to recover the true rank of $\mathbf{U_c}$ and $\mathbf{U_r}$ at the 30% and 60% noise levels for a threshold of $1e-3$ in the computation of the numerical rank. We also verified experimentally that the ℓ_1-norm penalty on $\mathcal{T}$ effectively promotes sparse solutions. We present in Figure 4.6 example plots for the non-zero elements of the first frontal slice of $\mathcal{T}$ and for the singular values of $\mathbf{U_c}$ and $\mathbf{U_r}$ recovered by *RPCA2D Fro* ℓ_1 on synthetic data with 30% noise.

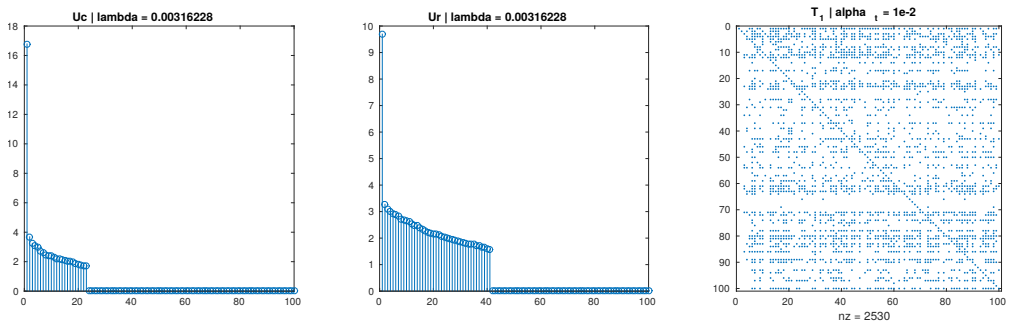

Figure 4.6: Rank recovery (as shown by the plot of the singular values) and sparsity of the core tensor for ℓ_1 regularisation and 30% noise

The method achieves a clear spectral gap at the desired ranks for both $\mathbf{U_c}$ and $\mathbf{U_r}$, and we confirm the efficiency of the Lasso regularisation to discard irrelevant coefficients from the core tensor. Moreover, additional experiments have shown that with the ℓ_1 penalty on $\mathcal{T}$ it is sometimes possible to achieve exact numerical zeros for the redundant columns of $\mathbf{U_c}$ and $\mathbf{U_r}$.

These observations generalise to the other methods for the rank, and to *RPCA2D GL* ℓ_1 for the sparsity of $\mathcal{T}$.

4.8 Discussion

In the previous sections, we formulated the non-orthogonal bi-dimensional robust approximation of the frontal slices of a tensor as an optimisation problem involving a relaxed instance of the *Tucker2* factorisation. We then presented several variants of the problem corresponding to different approaches to rank minimisation, and different levels of sparsity.

To summarise, we decompose a third-order tensor in four components:

- A *pair of dictionaries* $\mathbf{U_c}$ and $\mathbf{U_r}$, to be shared by all the frontal slices
- A *core tensor* $\mathcal{T}$ that contains the coefficients of each frontal slice represented as a combination of elements of the pair of dictionaries
- An *outliers* tensor $\mathcal{E}$

Our approaches to rank minimisation can be classified into two categories:

- *Spectral* approaches are those that penalise the rank of $\mathbf{U_c}$ and $\mathbf{U_r}$ by minimising a **function of their singular values**; in the present thesis we investigated the use of the Schatten-2 norm for its smoothness
- *Structural* approaches where the rank is indirectly penalised by minimising a **function of the matrix elements**; we enquired into the properties of the ℓ_1/ℓ_2 norm with each non-overlapping group chosen to be a column of the matrix in order to reduce the dimensionality of $\dim(\mathrm{Span}(\mathbf{U_c}))$ and $\dim(\mathrm{Span}(\mathbf{U_r}))$

Finally we examined two regularisation strategies for the core tensor $\boldsymbol{\mathcal{T}}$:

- Tikhonov or ℓ_2 regularisation is a *dense* regularisation strategy and has proved efficient at stabilising the methods numerically by penalising the Frobenius norm of each frontal slice of the tensor
- ℓ_1 regularisation is *sparse* and also stabilises the methods while promoting automatic elimination of the redundant coefficients in each frontal slice of $\boldsymbol{\mathcal{T}}$

Empirical evaluation on synthetic data has shown that all methods are able to recover the components with high accuracy, and that most can also preserve the structure - and even recover the **exact** number of non-zero coefficients in $\boldsymbol{\mathcal{E}}$ with the right choice of parameters.

In this section, we discuss additional aspects of our methods. We start by providing a mathematical justification to our choice of stopping criterion and show all four methods achieve linear convergence in agreement with the theoretical analysis of the ADMM algorithm. We then describe our initialisation procedure, as it is an important step considering the non-convexity of the problems we solve. Finally, we analyse the time and space complexity of our algorithms, discuss their scalability, and provide details on their numerical implementation and on possible extensions.

4.8.1 Convergence

The convergence criterion is a critical aspect of an iterative algorithm and shall not be overlooked. The classical choice as found in the literature is to compute the relative ℓ_2 reconstruction error on the whole observation tensor $\boldsymbol{\mathcal{X}}$, which in our case gives:

$$\epsilon = \frac{||\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{T}} \times_1 \mathbf{U_c} \times_2 \mathbf{U_r} - \boldsymbol{\mathcal{E}} - \mathbf{M}||_{\ell_2}}{||\boldsymbol{\mathcal{X}}||_{\ell_2}} \tag{4.16}$$

With $\mathbf{M} = \mathbf{0}$ if we do not chose to estimate the mean. This criterion is very widely applicable and makes sense from the point of view of the convergence of the sequence of the reconstructed tensors $\boldsymbol{\mathcal{T}} \times_1 \mathbf{U_c} \times_2 \mathbf{U_r} + \boldsymbol{\mathcal{E}} + \mathbf{M}$ to the true observed tensor $\boldsymbol{\mathcal{X}}$ as measured by the canonical norm on the right tensor space. The iterations then stop when $\epsilon < \delta$ so when the observed tensor is well-reconstructed as a whole.

In our case, however, we **explicitly** made the assumption that each individual frontal slice of $\boldsymbol{\mathcal{X}}$ carried information, so our goal is to obtain a high-quality decomposition of every slice individually. This leads us to define the convergence of our methods with respect to the **maximal reconstruction error** of the frontal slices of $\boldsymbol{\mathcal{X}}$:

$$\epsilon_{\text{m,slices}} = \max_{n=1,\dots,N} \left(\frac{||\mathbf{X}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}} - \mathbf{E}_n - \mathbf{M}||_{\text{F}}}{||\mathbf{X}_n||_{\text{F}}} \right)$$

And, when applicable, to the maximal error on the equality constraints introduced by splitting:

$$\epsilon_{\text{m,T}} = \max_{n=1,\dots,N} \left(\frac{||\mathbf{T}_n - \mathbf{K}_n||_{\text{F}}}{||\mathbf{T}_n||_{\text{F}}} \right) \quad \epsilon_{\text{m,bases}} = \max \left(\frac{||\mathbf{U_c} - \mathbf{A}||_{\text{F}}}{||\mathbf{U_c}||_{\text{F}}}, \frac{||\mathbf{U_r} - \mathbf{B}||_{\text{F}}}{||\mathbf{U_r}||_{\text{F}}} \right)$$

We can argue that this is a stronger convergence requirement; the algorithms stop when

$$\max(\epsilon_{\text{m,slices}},\ \epsilon_{\text{m,T}},\ \epsilon_{\text{m,bases}}) < \delta \tag{4.17}$$

so when *each* individual frontal slice is well-reconstructed and each splitting-induced constraint is satisfied.

Recall the general form of the problems we solve:

$$\begin{aligned} \min \quad & \alpha_c f(\mathbf{U_c}) + \alpha_r f(\mathbf{U_r}) + \textstyle\sum_n g(\mathbf{T}_n) + \sum_n \Omega(\mathbf{E}_n) \\ \text{s.t} \quad & \forall n, \mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} + \mathbf{E}_n \end{aligned} \tag{4.18}$$

The first order optimality conditions of problem (4.18) are:

$$\begin{aligned} &\forall n, \mathbf{X}_n - \mathbf{U_c^*}\mathbf{T}_n^*\mathbf{U_r^*}^\mathsf{T} - \mathbf{E_n^*} = 0 \quad \text{(primal feasibility)} \\ &0 \in \partial(\alpha_c f(\mathbf{U_c^*})) + \sum_n \langle \mathbf{\Lambda}_n, \frac{\partial}{\partial \mathbf{U_c}} h_n(\mathbf{X}_n, \mathbf{U_c}, \mathbf{U_r}, \mathbf{T}_n, \mathbf{E}_n)\rangle \\ &0 \in \partial(\alpha_r f(\mathbf{U_r^*})) + \sum_n \langle \mathbf{\Lambda}_n, \frac{\partial}{\partial \mathbf{U_r}} h_n(\mathbf{X}_n, \mathbf{U_c}, \mathbf{U_r}, \mathbf{T}_n, \mathbf{E}_n)\rangle \\ &\forall n, 0 \in \partial g(\mathbf{T}_n) + \langle \mathbf{\Lambda}_n, \frac{\partial}{\partial \mathbf{T}_n} h_n(\mathbf{X}_n, \mathbf{U_c}, \mathbf{U_r}, \mathbf{T}_n, \mathbf{E}_n)\rangle \\ &\forall n, 0 \in \partial\Omega(\mathbf{E}_n) + \langle \mathbf{\Lambda}_n, \frac{\partial}{\partial \mathbf{E}_n} h_n(\mathbf{X}_n, \mathbf{U_c}, \mathbf{U_r}, \mathbf{T}_n, \mathbf{E}_n)\rangle \end{aligned}$$

Where ∂ denotes the sub-differential (see [17, 18]) and h_n is the n^{th} equality constraint. Therefore our convergence criterion corresponds to asserting primal feasibility. In practice (4.16) and (4.17) are often relatively equivalent.

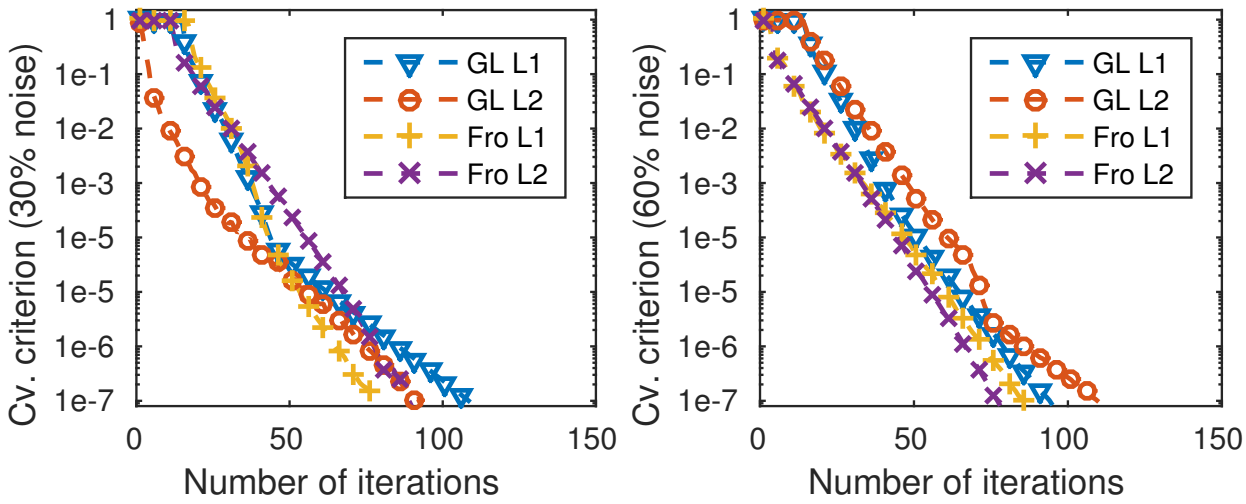

Figure 4.7: Empirical rate of convergence for 30% and 60% noise, on the run that yields the best relative reconstruction error

Figure (4.7) shows the convergence of the sequences generated by our four algorithms on the synthetic datasets used in section 4.7 represented by the maximum reconstruction error of the frontal slices plotted as a function of the number of iterations on a semi-logarithmic scale. Some algorithms begin with a short phase where the convergence criterion stays close to 1, then begin to converge at a linear rate until the target threshold ($1e-7$) is reached. We suggest that the initial dampened iterations are likely due to the strong requirement that the maximal reconstruction error be small.

4.8.2 Variable initialisation

Since our problems are non-convex, variable initialisation matters. The variables that need to be initialised are:

- The left and right bases $\mathbf{U_c}$ and $\mathbf{U_r}$
- The core tensor $\boldsymbol{\mathcal{T}}$
- The Lagrange multipliers $\boldsymbol{\Lambda}$
- The sparse outliers $\boldsymbol{\mathcal{E}}$
- The penalty parameters μ

We took a similar approach to that used in the *Robust PCA via inexact ALM (IALM)* code from Lin et al. [63] to initialise μ. Our strategy is to have

$$\mu^0 = \frac{\eta N}{\sum_n ||\mathbf{X}_n||_{\mathrm{F}}} \tag{4.19}$$

Where N is the total number of observations and η is a scaling coefficient. In the IALM code, the initialisation was done by setting $\mu^0 = 1.25/||\mathbf{X}||_{\mathrm{F}}$. Our idea is that since we chose the same μ for all the equality constraints of the form $\mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^{\mathsf{T}}$, we could replace the Frobenius norm of the observation matrix by the average of the Frobenius norms of all the data matrices. As for the value of η, we chose $\eta = 1.25$ as in [63].

Regarding the initialisation of $\mathbf{U_c}$, $\mathbf{U_r}$, and the $\mathbf{T}_n$ matrices; we compute an SVD for each frontal slice of $\boldsymbol{\mathcal{X}}$ such that $\forall n,\ \mathbf{X}_n = \mathbf{U}_n\mathbf{S}_n\mathbf{V}_n^{\mathsf{T}}$. Given an upper bound r on the rank of the solution, we then initialise the variables as follows:

$$\begin{aligned}
\mathbf{U_c} &= \frac{1}{N}\sum_n \mathbf{U}_n(:, 1:r) \\
\mathbf{U_r} &= \frac{1}{N}\sum_n \mathbf{V}_n(:, 1:r) \\
\mathbf{T}_n &= \mathbf{S}_n(1:r, 1:r)
\end{aligned}$$

Again, we average the effects of the observations to get an initial estimate of the left and right bases. However, we keep the singular values unaltered to preserve the specificity of each frontal slice.

We simply set the initial values of $\boldsymbol{\Lambda}$ and $\boldsymbol{\mathcal{E}}$ to zero even though other initialisation methods would have been acceptable. For instance, we could have first initialised $\mathbf{U_c}$, $\mathbf{U_r}$, $\boldsymbol{\mathcal{T}}$ and then used these values to set $\boldsymbol{\mathcal{E}}$:

$$\boldsymbol{\mathcal{E}}^0 = \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{T}}^0 \times_1 {\mathbf{U_c}}^0 \times_2 {\mathbf{U_r}}^0$$

Following Lin et al. [63] we update the sparse outliers first as it is empirically likely to promote convergence, so the initial value of $\boldsymbol{\mathcal{E}}$ is never read, and therefore any computation made for $\boldsymbol{\mathcal{E}}^0$ is lost.

In practice, our initialisation scheme gave satisfactory results in all the cases we tested.

4.8.3 Complexity analysis and scalability

Both the time and the space complexity of numerical algorithms are critical to their applicability. In the large-scale setting, memory is a valuable resource and should be used parsimoniously, while a high time complexity will lead to prohibitively high running times. On current desktop hardware, even relatively small problems can lead to excessive computation time and exhaust the available memory. Therefore, particular care was put in designing efficient numerical algorithms that can scale well to larger problems.

We report the asymptotic time and space complexity in O notation. For numerical algorithms, an interesting measure for real-world applications is the number of floating point operations per second, or *flops*. While it is certainly possible to give an estimate of the number of floating point operations required by each of our methods assuming the best possible algorithm was used for each operation, we found it unrealistic as the actual number is very implementation-dependent and can also influenced by the architecture of the computer on which the code is ran. Therefore, we give the asymptotic complexity as an indicator of how our methods can scale with the dimension of the problem.

We give an overview of the cost of our algorithms and comment on the scalability of our methods with respect to the dimensions of the observation tensor.

Complexity analysis

We give the cost in floating point operations and the storage required per iteration for Algorithm 1 - as it is at the basis of all of our methods - and the additional cost induced by each variant. We also discuss the time and storage required by our initialisation procedure. In order to keep this discussion concise and to allow us to focus on the interpretation of these results in terms of scalability, the detailed break-down of the analysis is presented in Appendix A.

We show that the cost in floating point operations (flop) and in space per iteration of the *base case* algorithm is:

$$O(N(mnr + (m+n)r + mn + \min(m,n)r^2 + r^3 + r^2)) \tag{4.20}$$

time, and

$$O(N(mn + (m+n)r + r^2)) \text{ or } O(Nmn + (m+n)r + r^2) \tag{4.21}$$

space, since the space complexity can be decreased at the expense of parallelism (see Appendix A).

Similarly, we determined that the additional cost induced by choosing an ℓ_1-norm penalty on $\mathcal{T}$ and solving the new problem by splitting is:

$$O(Nr^2) \text{ time and space} \tag{4.22}$$

On top of (4.20) and (4.21)

The additional cost of replacing the Frobenius norm on the bases by the Group Lasso comes from computing the proximal mapping of the ℓ_1/ℓ_2 norm and solving by splitting. We find it to be:

$$O(mr + nr) \text{ time and space} \tag{4.23}$$

On top of (4.20) and (4.21)

Finally, robustly estimating the mean at each iteration adds a cost of:

$$O(Nmn + mn) = O((N+1)mn) = O(Nmn) \text{ time and } O(Nmn) \text{ or } O(mn) \text{ space} \tag{4.24}$$

On top of (4.20) and (4.21)

In the worst case, our initialisation procedure involves computing a full SVD for each frontal slice of $\boldsymbol{\mathcal{X}}$ and averaging the N left-bases and N right-bases obtained. We also average the Frobenius norms of the frontal slices. We find that computing and averaging the SVDs yields a cost of:

$$O(N(mn\min(m,n) + (\min(m,n))^3 + mn) + mn)$$

time, and quadratic space per frontal slice. Averaging the N Frobenius norms has a cost of:

$$O(Nmn) \text{ time and } O(N) \text{ space}$$

Scalability

Noting that $r \leq \min(m,n)$, several observations can be made on the costs (4.20) and (4.21):

- The costs are linear in N, therefore the time and space taken by the method scale linearly with the number of frontal slices
- Both polynomials are homogeneous in m and n and the time depends on the minimum of the two, thus there is some advantage in having observations of rectangular shape, such as *tall & skinny*, or *short & fat*
- The running time increases cubically, and the storage required, quadratically with r. Thus increasing the upper bound on the n-ranks of $\boldsymbol{\mathcal{T}}$ yields - relatively - the largest increase in time and space costs. This emphasises the importance of solving the Stein equations encountered in the updates of $\boldsymbol{\mathcal{T}}$ in an efficient way; the naive way - *i.e,* by vectorising the equation - would give a prohibitive time cost of $O(r^6)$ and space cost of $O(r^4)$, and make our methods impractical

It is also clear from (4.22), (4.23), and (4.24) that the additional complexity of the proposed variants of Algorithm 1 is equivalent to or dominated by the complexity of the base algorithm, so the same discussion on scalability applies.

The initialisation procedure for $\boldsymbol{\mathcal{T}}$ and the bases is linear in N but scales cubically in the dimensions of the frontal slices, which is a concern for large matrices. On the other hand, initialising μ only requires a number of floating point operations linear in each dimension of $\boldsymbol{\mathcal{X}}$ individually - and thus linear in the total number of elements of $\boldsymbol{\mathcal{X}}$ - so should not be a major obstacle when scalability is important.

4.8.4 Implementation details

We used Matlab 2015a [71] because this scientific package is designed for linear algebra-heavy algorithms and is the package used by most authors in the field. This allowed us to re-use the code provided by the authors for our experimental comparisons, avoiding the pitfalls of writing our own implementations in a different language.

We made use of several toolboxes and extensions:

- We used the Image Processing Toolbox for image visualisation and adding salt & pepper noise
- The `dlyap` routine comes from the Optimisation and Control toolbox
- In order to have fast tensor mode-n product, we used the MMX extension [96], written in C++ with OpenMP, to parallelise the work along the dominant dimension of the tensor

- For compatibility purposes with the work of other authors, we included the Matlab Tensor Toolbox [8] and the Tensorlab [102] package
- We wrote some extensions ourselves in C with BLAS/LAPACK and OpenMP for speeding-up some bottleneck computations

Efficient computation of the SVD

We mentioned in our discussion on scalability that computing expensive SVDs for the initialisation procedure can become a concern when the frontal slices of the observation tensor are large. We suggest investigating more efficient algorithms for computing the SVD, especially when a bound on r is known and much smaller than the dimensions of the matrix, or when the matrix has a particular structure that provides a computational advantage.

When $r < \min(m, n)$ we can avoid computing a full SVD as we only need to compute the first r singular vectors and singular values. This can be done faster than a full SVD, especially if the matrix is sparse [58].

Other options include the more recent randomised SVD (Halko et al. [46]), suitable for large datasets.

Extension to sparse numerical linear-algebra

As we mentioned before, sparse representations could be adopted for some of the elements of our decompositions. The clearest example is the *sparse outliers* tensor $\boldsymbol{\mathcal{E}}$, for which a sparse array would be natural. We experimentally observed that ℓ_1 regularisation on $\boldsymbol{\mathcal{T}}$ effectively produces sparse results in most cases, so $\boldsymbol{\mathcal{T}}$ could also be stored as a sparse array. Finally, sparse representations could be investigated for the bases, especially in the case of structural rank-minimisation. Depending on the total size of $\mathbf{U_c}$ and $\mathbf{U_r}$ and on the number of non-zero columns found, the arrays could be considered sparse in practice.

The issue, however, is that sparse representations would most likely be less efficient than plain arrays when the arrays are not sparse. Therefore, since the aforementioned variables do not necessarily assume sufficient sparsity levels in the early stages of the optimisation procedure, a possible solution would be to start with plain arrays and switch to sparse representations when the sparsity level of a given matrix or tensor is below a pre-defined threshold that *depends on the structure of the matrix and on the operations we wish to perform on it* [39].

Sparse linear algebra has many advantages over traditional linear algebra for sparse arrays. The memory requirements can be drastically reduced and, *in general*, the time complexity of linear algebra algorithms becomes proportional to the number of non-zero entries of the array.

Extension to distributed computing

We argue that our methods can be made more scalable by distributing the computations to multiple workers. In fact, we can highlight three key points regarding parallelism. We base our analysis on Algorithm 1 and mention the extensions when relevant:

- Matrix-matrix multiplications, matrix decompositions, tensor addition and scaling as well as proximal operators such as the soft-shrinkage operator, or the proximity operator of the ℓ_1/ℓ_2 norm can all be implemented in a parallel fashion for a variety of parallel architectures. Our implementation already makes use of parallel BLAS and LAPACK libraries when available, and parallel algorithms exist for many linear-algebra operations. This helps scaling the problem to **large frontal slices**

- Several steps in our algorithms involve computing the *reduction* of a collection of elements with respect to a given operator. This is mostly the case when we compute the sum of N matrices or matrix products. These operations can naturally be performed in a *MapReduce* way: given a collection of K workers, the N matrix products can be split in K groups and each assigned to a worker. One master node would then collect the results and perform the summation (*i.e,* the reduction). This helps scaling the problem to a **large number of frontal slices**
- Updating $\mathbf{T}_n$ for a given n is independent of the values of $\mathbf{T}_k,\ k \neq n$. This makes the update of $\mathcal{T}$ *embarrassingly parallel*. Depending on the representation adopted for $\mathcal{T}$, either as a collection of individual slices or as a single contiguous array, the problem can also be seen as an instance of the *MapReduce* pattern where the reduction operator is the concatenation of the $\mathbf{T}_n$ matrices in a single tensor. This also helps scaling the problem to a **large number of frontal slices**

4.8.5 Relation to sparse dictionary learning

Sparse dictionary learning learns sparse representation of data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ (*i.e,* coding) as linear combinations of dictionary *atoms* $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_n]$ such that the representation $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_N]$ is sparse. Mathematically, the optimisation problem is:

$$\underset{\mathbf{D},\mathbf{R}}{\operatorname{argmin}} \sum_i ||\mathbf{x}_i - \mathbf{D}\mathbf{r}_i||_2^2 + \lambda||\mathbf{r}_i||_0 \tag{4.25}$$

Contrary to PCA, the atoms are not required to be orthogonal, and the representation space can be of higher dimensionality than the input space. We refer to Murphy [73] for an introduction to the subject.

Let us consider our method with ℓ_1 penalty on $\mathbf{T}_n$ and Frobenius penalties on $\mathbf{U_c}$ and $\mathbf{U_r}$. The model is:

$$\mathbf{X}_n = \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} + \mathbf{E}_n$$

Vectorising both sides of the equation we obtain:

$$\operatorname{vec}(\mathbf{X}_n) = (\mathbf{U_r} \otimes \mathbf{U_c})\operatorname{vec}(\mathbf{T}_n) + \operatorname{vec}(\mathbf{E}_n)$$

Recall the form of the Augmented Lagrangian, each sub-problem consisted in a penalised least-squares. Introducing the constraints on $\mathbf{T}_n$, $\mathbf{U_c}$, $\mathbf{U_r}$ we get the sub-problem for $\mathbf{T}_n$:

$$\min ||\operatorname{vec}(\mathbf{X}_n) - (\mathbf{U_r} \otimes \mathbf{U_c})\operatorname{vec}(\mathbf{T}_n)||_2^2 + ||\operatorname{vec}(\mathbf{T}_n)||_1$$

Which is exactly a sparse dictionary-learning problem.

Assuming $||\mathbf{A} \otimes \mathbf{B}||_\mathrm{F} = ||\mathbf{A}||_\mathrm{F}||\mathbf{B}||_\mathrm{F}$, since we have $||\mathbf{A}||_\mathrm{F}||\mathbf{B}||_\mathrm{F} \leq \frac{1}{2}(||\mathbf{A}||_\mathrm{F}^2 + ||\mathbf{B}||_\mathrm{F}^2)$ we find the variational characterisation of the nuclear norm of $\mathbf{A}\mathbf{B}^\mathsf{T}$. More research is needed based on this assumption.

4.9 Summary

Algorithm	Optimisation objective	Resolution	Stopping criterion	Time complexity	Space complexity
Fro ℓ_2 (base)	$\frac{\alpha_c}{2}\|\|\mathbf{U_c}\|\|_{\mathrm{F}}^2 + \frac{\alpha_r}{2}\|\|\mathbf{U_r}\|\|_{\mathrm{F}}^2$ $+\frac{\alpha_t}{2}\sum_n \|\|\mathbf{T}_n\|\|_{\mathrm{F}}^2 + \|\|\boldsymbol{\mathcal{E}}\|\|_1$	ADMM	$\epsilon_{\mathrm{m,slices}} < \delta$	$O(N(mnr + (m+n)r + mn + \min(m,n)r^2 + r^3 + r^2))$	$O(N(mn + r^2 + (m+n)r))$
Fro ℓ_1	$\frac{\alpha_c}{2}\|\|\mathbf{U_c}\|\|_{\mathrm{F}}^2 + \frac{\alpha_r}{2}\|\|\mathbf{U_r}\|\|_{\mathrm{F}}^2 +$ $\alpha_t \sum_n \|\|\mathbf{T}_n\|\|_1 + \|\|\boldsymbol{\mathcal{E}}\|\|_1$	ADMM + Splitting	$\max(\epsilon_{\mathrm{m,slices}},\ \epsilon_{\mathrm{m,T}})$ $< \delta$	Base + $O(Nr^2)$	Base + $O(Nr^2)$
GL ℓ_2	$\alpha_c\|\|\mathbf{U_c}\|\|_{\ell_1/\ell_2} + \alpha_r\|\|\mathbf{U_r}\|\|_{\ell_1/\ell_2}$ $+\frac{\alpha_t}{2}\sum_n \|\|\mathbf{T}_n\|\|_{\mathrm{F}}^2 + \|\|\boldsymbol{\mathcal{E}}\|\|_1$	ADMM + Splitting	$\max(\epsilon_{\mathrm{m,slices}},\ \epsilon_{\mathrm{m,bases}})$ $< \delta$	Base + $O((m+n)r)$	Base + $O((m+n)r)$
GL ℓ_1	$\alpha_c\|\|\mathbf{U_c}\|\|_{\ell_1/\ell_2} + \alpha_r\|\|\mathbf{U_r}\|\|_{\ell_1/\ell_2}$ $+\alpha_t \sum_n \|\|\mathbf{T}_n\|\|_1 + \|\|\boldsymbol{\mathcal{E}}\|\|_1$	ADMM + Splitting	$\max(\epsilon_{\mathrm{m,slices}},\ \epsilon_{\mathrm{m,T}},$ $\epsilon_{\mathrm{m,bases}}) < \delta$	Base + $O(Nr^2 + (m+n)r)$	Base + $O(Nr^2 + (m+n)r)$
Any with robust estimation of the mean	—	Additional step $\hat{\mathbf{M}}^t = \frac{1}{N}\sum_n(\mathbf{X}_n - \mathbf{E}_n^t)$	—	Add $O(Nmn)$	Add $O(Nmn)$

Table 4.3: Summary of our methods

Chapter 5

Comparison with the state of the art

In section 4.7 we validated our approach experimentally by showing that our models effectively separate the low-rank component from the sparse outliers on synthetic data designed to challenge the models' assumptions, and that exact recovery was attainable for both the low-rank and sparse components of the tensor even at high noise levels. We also showed the validity and limitations of our iterative procedure for robust estimation of the mean by comparing the recovered mean to the original sample mean. Our experiments also demonstrated that the true ranks along the first and second modes of the low-rank tensor is discovered by all four methods.

In this chapter, we evaluate the empirical performance of our models on real data as well as their performance compared to the current state of the art. We include both convex and non-convex optimisation-based algorithms as well as robust Bayesian tensor decompositions. We report the performance of the algorithms on a set of popular computer vision benchmarks chosen to cover different use cases. We show that our methods consistently outperform or match the performance of current state-of-the-art algorithms according to five image-quality measures covering three key aspects of image reconstruction: pixel-wise noise reduction, preservation of the structural information, and preservation of the spectral information.

5.1 Experimental setup

Let us first give a detailed description of our experimental methodology. Our goal is to produce as fair and accurate as a comparison possible, while keeping the testing conditions realistic to test each algorithm's applicability to real-world computer vision tasks. We also emphasise the reproducibility of the results by giving explicit values for the parameters used for each algorithm in Appendix B. Images are kept at their original sizes and pixel intensities are rescaled to lie in the $[0, 1]$ interval.

5.1.1 Algorithms compared

We chose to compare the following algorithms:

- Non-orthogonal 2D RPCA ℓ_2-regularisation on the core
- Non-orthogonal 2D RPCA ℓ_1-regularisation on the core
- Non-orthogonal 2D RPCA with Group Lasso and ℓ_2-regularisation on the core
- Non-orthogonal 2D RPCA with Group Lasso and ℓ_1-regularisation on the core
- Tensor RPCA (Papamakarios et. al. 2014)

- Tensor BRPCA (Papamakarios et. al. 2014)
- Tensor RHOSVD (Papamakarios et. al. 2014)
- Tensor RCPD with substitution (Papamakarios et. al. 2014)
- Tensor RPCA via TNN (CVPR2014)
- Tensor RPCA via TNN (CVPR2016)
- Non-convex Tensor RPCA (Anandkumar et. al. 2016)
- HORPCA-S (Goldfarb & Qin 2013)
- Bayesian Robust Tensor Factorisation (BRTF 2016)
- Robust Tensor Factorisation via redescending M-Estimators - Welsh loss and soft-thresholding (2015)
- Robust Tensor Factorisation via redescending M-Estimators - Cauchy loss and soft-thresholding (2015)

The panel of methods tested spans several years, and all major approaches to tensor RPCA are represented.

Tensor RPCA from Papamakarios [76], and *HORPCA-S* from Goldfarb and Qin [40] are convex optimisation methods and work on the tensor mode-n unfoldings. *Tensor BRPCA* [76] uses the variational characterisation of the nuclear norm (Theorem 2.1.15) on the tensor unfoldings to avoid expensive SVD computations. Other methods based on the mode-n matricisations we tested employ *redescending M-estimators* (Yang et al. [110]), we chose to start by including the soft-thresholding version of the proposed algorithms for the two loss functions the authors investigate: namely the *Cauchy loss* and the *Welsh loss*. Two robust factorisations are part of the study: *Tensor RHOSVD* [76] is a robust Tucker decomposition while *Tensor RCPD* [76] is a robust CP. The recent methods derived from the *Tensor Nuclear Norm* defined in terms of the frontal slices in the Fourier domain are represented through two papers, the first one from 2014 was accepted for oral presentation at CVPR and can be considered foundational work for this approach. The more recent reference from CVPR 2016 [66] builds upon [115] with subtle but important differences. One non-convex method has been included, and we chose to this effect the very recent work of Anandkumar et al. [4] and may chose to test the non-convex version of *HORPCA* from Goldfarb & Qin [40], or the hard-thresholding versions of the redescending M-estimator methods in subsequent work. Finally, we couldn't get code for robust methods based on the tensor spectral norm and leave there inclusion to future work.

Regarding Bayesian models, we only incorporated *BRTF* [118] for several reasons. The first being the relative rarity of such models compared to optimisation-based models which inherently limits the number of candidate algorithms. Moreover, we initially planned to include the model from Chen et al. [25], but problems with the implementation provided by the authors - such as our impossibility to reproduce the results from the paper in spite of following the authors' directions, and the long execution time of the method - drove us to withdraw the model from the study, at least until the experimental issues can be fixed. Finally, one of the possible goals we initially considered for this thesis was to develop a robust Bayesian CP factorisation, which is precisely what *BRTF* achieves.

5.1.2 Parameters tuning

We made the **deliberate choice** of tuning as few parameters as possible for each method. Such a decision is motivated by the fact that in most applications, fine-tuning many parameters to achieve good performance is too time-consuming to be realistically expected of the user. We believe the practical applicability of a method is highly dependent on its ability to perform consistently well with a default choice for all but one or two parameters, so we kept the default values as set by the authors of the implementations tested and tuned at most two parameters by standard grid search. Table (5.1) describes which parameters were tuned in all benchmarks. Special cases will be explicitly mentioned in the benchmarks' own sections.

For methods that provide a rule for choosing the optimal value of one parameter (*e.g,* [115, 66, 40, 110] give a closed-form expression for an optimal - or empirically effective - choice for λ, which will be denoted λ^*), we built the search interval by first computing which powers of ten bound the optimal value from above and from below, *i.e,* we find integers p and q with $p < q$ such that $10^p < \lambda^* < 10^q$. The final set consists of 30 or 40 points linearly sampled from $[\frac{1}{2}10^p,\ \frac{3}{2}10^q]$.

In the case of [110], we based our search range on the authors' guidelines for choosing λ. The choice of σ was more difficult and we decided to test a wide interval via grid-search. Appendix E of [40] also provides rules for setting suitable parameters for *HORPCA-S*, we again followed the directives while extending the search range to accommodate the diversity of our benchmarks.

In the case of *BRTF*, the initialisation value was the only parameter we could tune since the very point of Bayesian methods is to infer optimal parameters from the data by treating the unknown parameters as standard hidden variables of the model. We sampled 30 points on a logarithmic scale spanning a wide range of values.

The case of non-convex methods is different because these methods expect a value for the rank r in the rank constraint as shown by the below generic form of non-convex matrix low-rank optimisation problem:

$$\min \mathcal{F}(\mathbf{X}) \quad \text{s.t} \quad \text{rank}(\mathbf{X}) \leq r$$

r needs to be chosen carefully since the algorithm is not able to determine its value automatically. One possible approach is to first train a convex model to get an estimate of then rank, and then feed this information to the non-convex model. In our case, however, we were also interested in the evolution of the performance with respect to the value of r, so we tried several suitable values.

Finally for both the 2D methods and the methods introduced in [76]. we chose a range of values for λ that spans the values given by criteria similar to the original Robust PCA criterion, such as $\lambda = \frac{1}{\sqrt{n_1 n_2 n_3}}$ or $\lambda = \frac{1}{\sqrt{n_3 \max(n_1, n_2)}}$. In this case, we chose to sample 30 values equally spaced on a logarithmic scale between $2e^{-4}$ and $2e^{-1}$.

Note: Our 2D methods, *BRTF*, and the methods introduced in [76] can incorporate information on an upper bound on the hidden ranks. This is likely to improve the performance when the bound is close to the maximum true rank. We therefore emphasise that in these experiments, no such bound was provided for generality.

Table (5.2) gives a summary of the parameter ranges per algorithm. According to the benchmarks, we either enable or disable the iterative robust estimation of the mean in our 2D algorithms if it improves the performance.

Algorithm	First parameter tuned	Second parameter tuned
Tensor RPCA	λ	
Tensor BRPCA	λ	
Tensor ORPCA	λ	
Tensor RCPD	λ	
2D ℓ_2	λ	
2D ℓ_1	λ	
2D GL ℓ_2	λ	
2D GL ℓ_1	λ	
Tensor RPCA (CVPR2014)	λ	
Tensor RPCA (CVPR2016)	λ	
NC TRPCA	Threshold constant	Rank
HORPCA-S	λ	Zeros as missing values? (S&P noise)
BRTF	Initialization constant	
Welsh ST	σ	λ
Cauchy ST	σ	λ

Table 5.1: The algorithms tested and the parameters we tuned for each of them

5.2 Background subtraction

The goal of background subtraction is to separate the background from the foreground in a sequence of images - typically a video. Background subtraction has various applications, most notably to video surveillance to make the automated detection of suspicious activity easier.

In the case of a static background with moving objects, the tensor formed from the backgrounds of the frames will naturally be low-rank, while the foreground can be modelled as a sparse component containing the changing elements that are *outliers* with respect to the empirical distribution of the pixels of the static background. Hence, subtracting a static background can be seen as low-rank modelling problem. A more challenging task is modelling moving or changing backgrounds. A commonly encountered example is that of a video sequence recorded outdoors where the background is unlikely to be static due to small random changes such as moving tree branches and tree leaves, people walking, or waves on the sea. We tested both cases through two different experiments.

We will treat the background subtraction problem as a binary classification exercise: given a pixel x in one frame, does x belong to the background or to the foreground?

In the two experiments we present, either whole or part of the correct classification is known. This forms the *Ground Truth* of the dataset. Measuring the performance of the classifiers boils down to measuring the similarity between the Ground Truth and the assigned classes. A classical performance metric for binary classification is the *Area Under the Curve (AUC)* which is defined as the area contained under the *Receiver Operating Characteristic (ROC)* curve [35]. The ROC curve is a graphical representation of the effectiveness of a binary classifier; given a *positive* class *P* and a *negative* class *N*, the ROC is obtained by plotting the fraction of correctly classified samples for the positive class (True Positive Rate) as a function of the fraction of samples incorrectly classified as belonging to *P* (False Positive Rate). Equivalently, we can represent the False Negative Rate as

Algorithm	**Range for first parameter**	**Range for second parameter**
Tensor RPCA	$2\text{logspace}(-4,-1,30)$	
Tensor BRPCA	$2\text{logspace}(-4,-1,30)$	
Tensor ORPCA	$2\text{logspace}(-4,-1,30)$	
Tensor RCPD	$2\text{logspace}(-4,-1,30)$	
2D ℓ_2	$2\text{logspace}(-4,-1,30)$	
2D ℓ_1	$2\text{logspace}(-4,-1,30)$	
2D GL ℓ_2	$2\text{logspace}(-4,-1,30)$	
2D GL ℓ_1	$2\text{logspace}(-4,-1,30)$	
Tensor RPCA (CVPR2014)	$\text{linspace}(0.5ep, 1.5eq)$ $10^p < \frac{1}{\sqrt{\max(n_1,n_2)}} < 10^q$	
Tensor RPCA (CVPR2016)	$\text{linspace}(0.5ep, 1.5eq)$ $10^p < \frac{1}{\sqrt{n_3 \max(n_1,n_2)}} < 10^q$	
NC TRPCA	$[10, 1e2, 1e3, 1e4]$	Rank depending on $\min(n_1, n_2)$
HORPCA-S	$\text{logspace}(-1,1,30)\frac{1}{\sqrt{\max(n_1,n_2,n_3)}}$	Zeros as missing values? (S&P noise)
BRTF	$\text{logspace}(-7,1,30)$	
Welsh ST	$\text{logspace}(\log_{10}(0.05), \log_{10}(2), 20)$	$\lambda = \alpha\frac{\min(n_1,n_2)}{\sqrt{\max(n_1,n_2)}}$ $\alpha \in$ $\text{linspace}(0,1,40)$
Cauchy ST	$\text{logspace}(\log_{10}(0.05), \log_{10}(2), 20)$	$\lambda = \alpha\frac{\min(n_1,n_2)}{\sqrt{\max(n_1,n_2)}}$ $\alpha \in$ $\text{linspace}(0,1,40)$

Table 5.2: Range of parameters tested for each algorithm in Matlab notation

a function of the False Positive Rate and obtain the *Detection Error Trade-off* curve (Martin et al. [68]). We will use this representation instead of the ROC as it has a very clear interpretation: the closer the curves are to the origin, the higher the performance. The AUC metric ranges from 0.5 to 1; a classifier with an AUC of 0.5 doesn't perform better than random assignment of the classes and is therefore of no practical use. On the contrary, an AUC of 1 denotes a perfect classifier.

For all background subtraction experiments, we enabled the robust estimation of the mean for our methods as experiments have shown it is of great benefit in this case. This is easily explained by the fact that when the background is static, the background of every frame will be very close - if not equal - to the mean. Moreover, the case of a sparse component of moving objects is favourable to our estimation procedure since we can expect each area of the true mean to stay unobstructed often enough to be correctly estimated.

5.2.1 The highway dataset

Our first benchmark is the *highway* dataset, which consists in an excerpt of a video sequence of cars travelling on a highway. We reuse the experiment from Papamakarios [76] and keep the data as 400 gray-scale images of resolution 48×64. This dataset, adapted from [42], is well suited to our

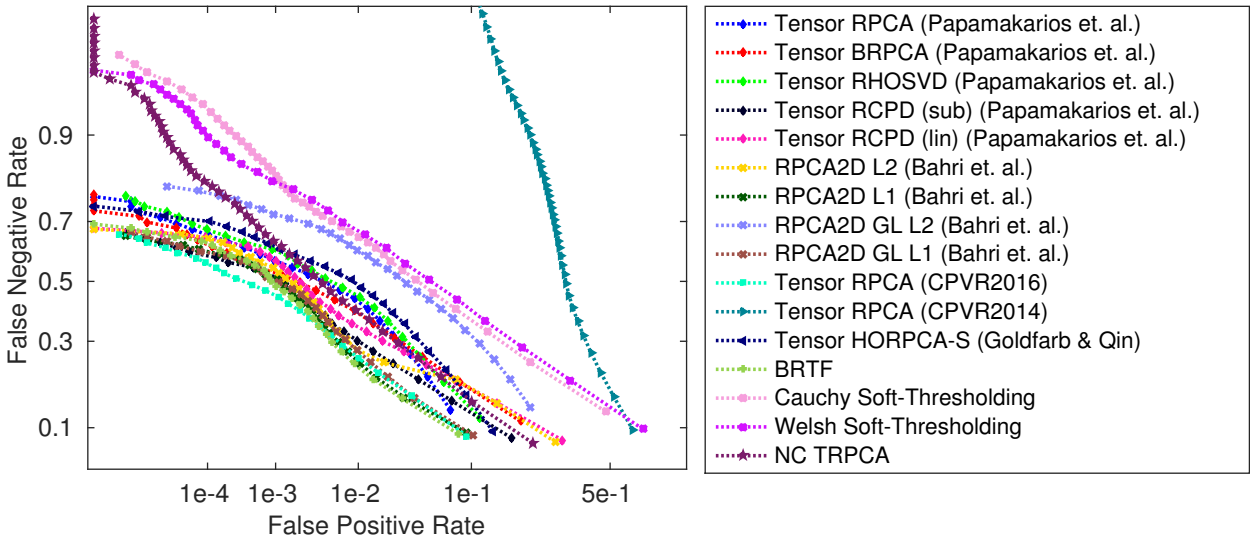

Figure 5.1: DET curves on the *Highway* dataset

test for several reasons; the small size of the images allows for reduced computation time, but more importantly, the background is completely static and hence the tensor formed of the backgrounds of the frames will be low-rank.

We tested the algorithms with the range of parameters described in Table 5.2, we report their AUC scores in Table 5.3 and the DET curves in Figure (5.1). Additionally, the bounds on the rank required by the methods from [76] were set exactly as in the original experiment for accurate comparison.

As a comparison we give Table (5.4) taken from Papamakarios [76] that includes previous evaluation of some tensor and matrix methods. The experimental results are clear: the best-performing matrix methods outperform the best-performing tensor methods. Our new experiments show that the current best tensor methods have caught up on this particular benchmark and now perform equally well as the best matrix methods. It should also be noted that while certain matrix methods require us to provide an upper bound on the rank, the best-performing tensor methods do not require this information.

Regarding the speed of the various algorithms measured both in terms of time and number of iterations taken, we would like to emphasise that one should be very cautious in drawing any conclusion from the results reported. Background subtraction benchmarks are not designed to truly and accurately assess the speed of the different methods; moreover, we distributed the work on different machines that were not dedicated to our experiments, so the general usage level of the machines can impact the running time. More importantly, not all implementations are of equal degree of sophistication, and inefficiencies in an algorithm's implementation can increase both its time and memory requirements. Finally, the number of iterations taken by iterative algorithms depends directly on the convergence threshold set by the user. We set the convergence threshold of our methods to $1e-5$ thanks to experiments showing further iterations were not beneficial, but the algorithms would have ran for longer or shorter had we set the threshold differently. **In summary, the time and number of iterations we report are only for indicative purposes.**

To conclude this experiment, we show in Figures 5.2 and 5.3 two frames and their ground truth along with the recovered backgrounds and foregrounds for visual assessment of the results.

Algorithm	**AUC**	**Parameter 1**	**Parameter 2**	**Iterations (indicative)**	**Time (indicative)**
BRTF	0.9451	0.0000161	100	118	18.40
Tensor RPCA (CPVR2016)	0.9449	0.00156		213	137.81
RPCA2D GL ℓ_1	0.9432	0.00255	0.0464	50	66.94
RPCA2D Fro ℓ_1	0.943	0.00367		36	51.25
Tensor RCPD (sub)	0.936	0.001	50	83	48.46
NC TRPCA	0.9292	48	1000	99	357.39
Tensor HORPCA-S (Goldfarb & Qin)	0.925	0.0245		58	65.49
RPCA2D Fro ℓ_2	0.9241	0.00367		33	46.87
Tensor RHOSVD	0.9202	0.01	0.8	89	53.62
Tensor RPCA	0.9195	0.01		76	47.95
Tensor RCPD (lin)	0.9192	0.0316	50	82	47.17
Tensor BRPCA	0.9145	0.0316	0.6	92	20.16
RPCA2D GL ℓ_2	0.8745	0.000595	0.1	27	38.41
Cauchy Soft-Thresholding	0.8324	3.08	0.0737	501	71.82
Welsh Soft-Thresholding	0.8188	6	0.236	501	64.54
Tensor RPCA (CPVR2014)	0.614	0.0563		1000	669.73

Table 5.3: Classification performance on the *Highway* dataset in descending order of AUC

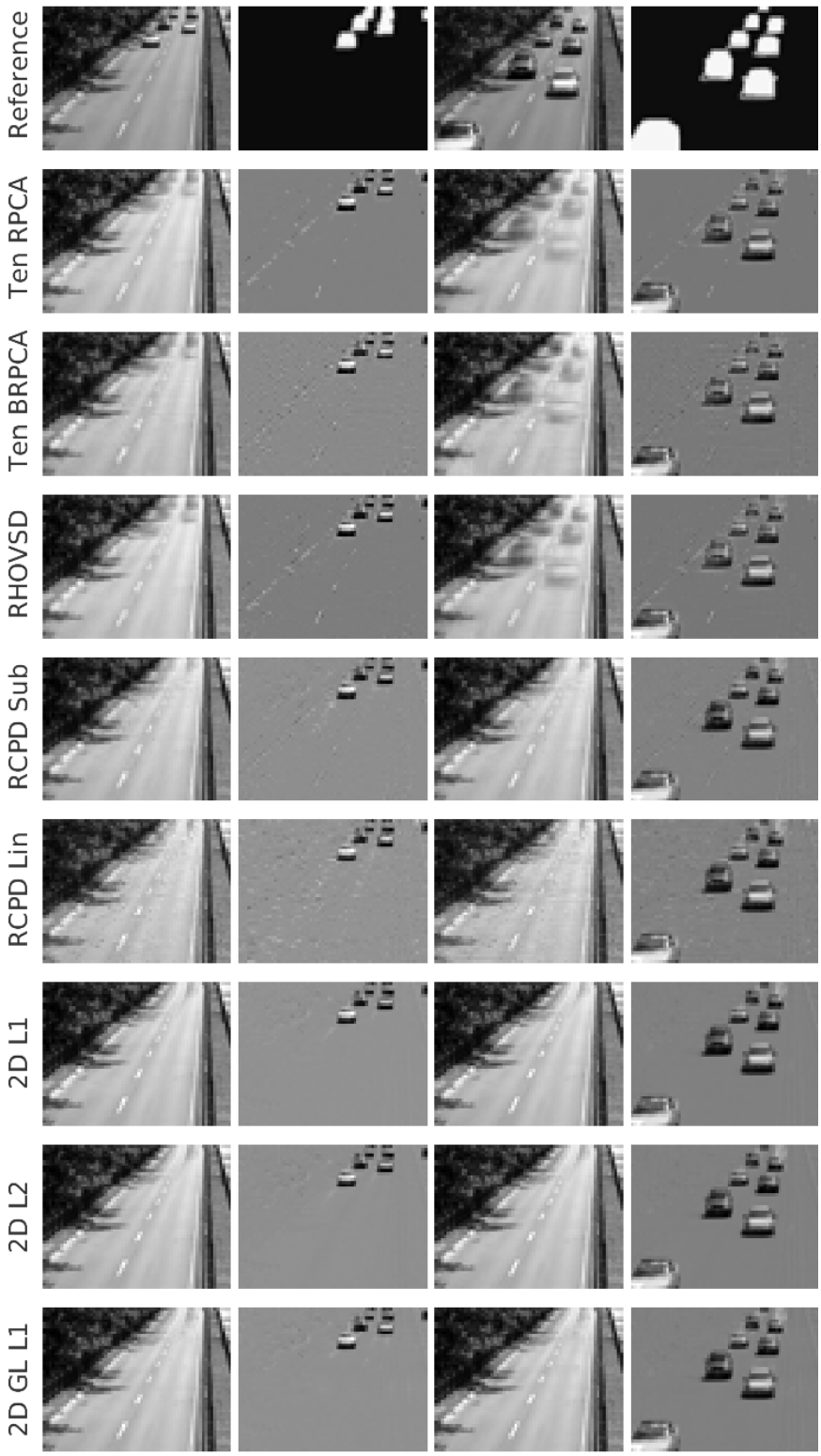

Figure 5.2: Background subtraction results on the *Highway* dataset - part 1

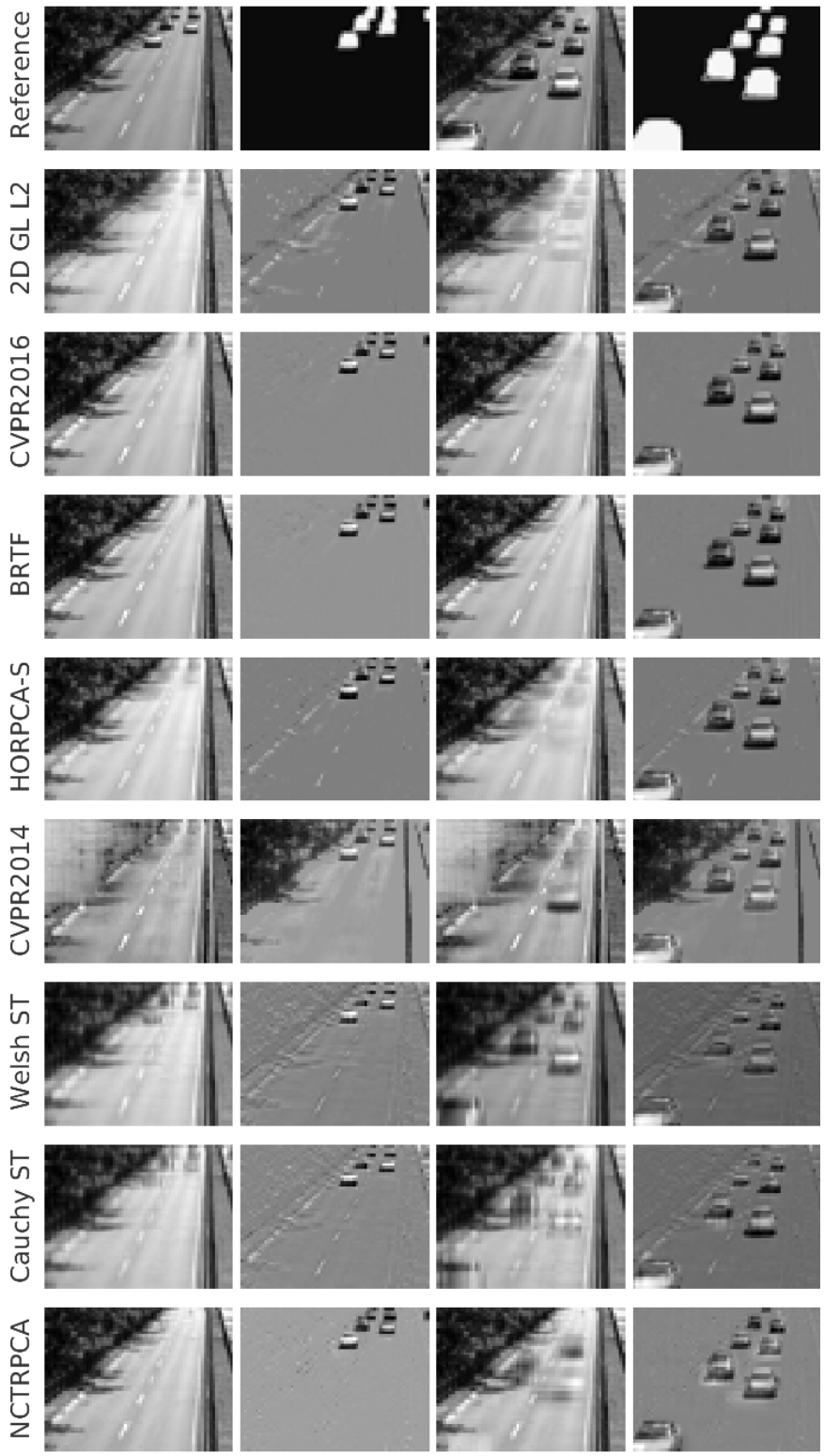

Figure 5.3: Background subtraction results on the *Highway* dataset - part 2

	Algorithm	AUC	Time [sec] (indicative)	Iterations (indicative)	λ	r
Matrices	RPCA (alm)	0.944	13.658	85	0.0100	–
	RPCA (apg)	0.944	30.138	183	0.0100	–
	BRPCA	0.945	3.430	82	0.0100	10
	IRPCA (sub)	0.918	1347.867	92	0.0001	–
	IRPCA (lin)	0.929	935.764	74	0.0100	–
	ORPCA	**0.946**	**5.737**	**88**	**0.0316**	**30**
	ROSL	0.944	16.590	58	0.0100	50
Tensors	RPCA	0.920	52.136	76	0.0100	–
	BRPCA	0.915	19.034	91	0.0316	0.6×
	IRPCA (sub)	0.884	34.748	85	0.0010	0.2×
	IRPCA (lin)	0.814	35.028	83	0.1000	0.2×
	RHOSVD	0.921	52.495	89	0.0100	0.8×
	RCPD (sub)	**0.937**	**37.039**	**82**	**0.0010**	**40**
	RCPD (lin)	0.921	46.206	88	0.1000	50

Table 5.4: Numerical result for background subtraction on the highway dataset adapted from Papamakarios [76], the best-performing matrix methods are shown to outperform the best-performing tensor methods

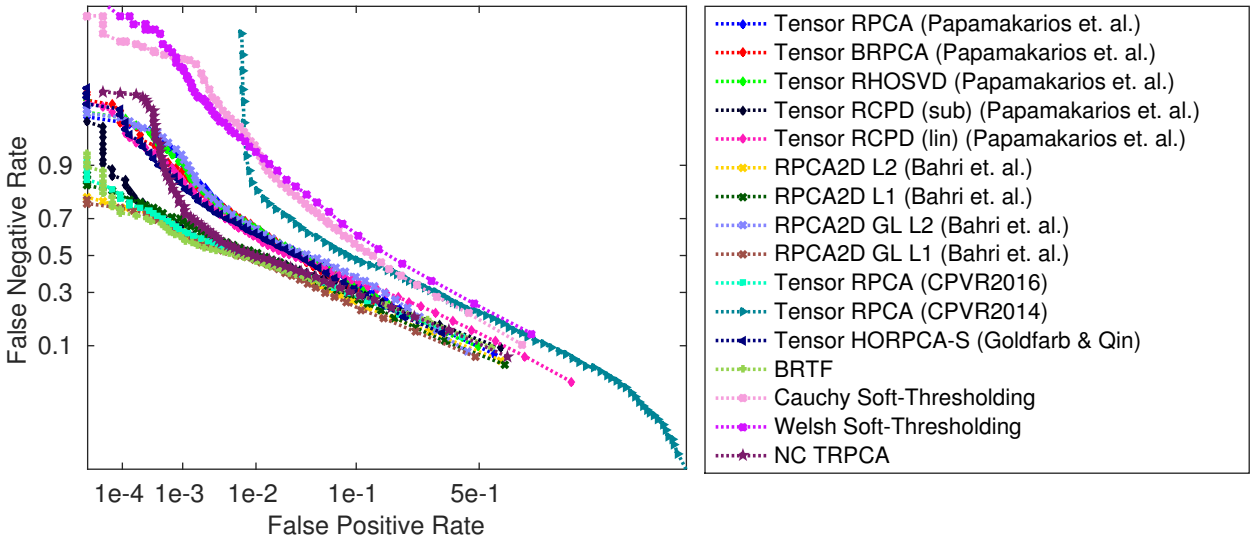

Figure 5.4: DET curves on the *Airport Hall* dataset

5.2.2 The airport hall dataset

The second dataset we used in our background subtraction experiments is the *Airport Hall* dataset (Li et al. [60]). Only a partial ground truth is available, and there are too many frames in the original video for our purpose. We use the same excerpt of 300 frames (from frame 3301 to 3600) as in Zhao et al. [118] and keep the frames to their original sizes. Two ground truth frames are available for the range we chose and are used for performance assessment. The experimental methodology is the same as for the *Highway dataset*, we measure the AUC on the ground truth for a range of parameters described in Table 5.2 and plot the *Detection Error Trade-off* curve in Figure 5.4.

The best performing 2D methods are again *RPCA2D GL* ℓ_1 and *RPCA2D Fro* ℓ_1, closely followed by *RPCA2D Fro* ℓ_2. The Group Lasso with Tikhonov regularisation still achieves over 0.85 AUC, but it is significantly less than the three other methods. Interestingly, the best non-2D algorithm is also based on the frontal-slices and is the recent *Tensor RPCA* presented in CVPR 2016 [66]. The non-convex *NC TRPCA* also performs well and ranks 5^{th} on this benchmark.

BRTF, which was the best method on the *Highway* benchmark is not among the five-best algorithms but demonstrates good performance nonetheless and visually pleasing results as shown in Figure 5.6. Excluding the people standing still at the desk, *BRTF* is the method with the less contamination between the background and the foreground. However, the ground truth includes these people in the foreground, so methods with more contamination but also more pixels from this area in the foreground fare better according to the AUC metric.

Visually, the three well-performing 2D models achieve the best trade-off between detection of the immobile people and background-foreground contamination. Understandably, their AUC scores are markedly higher than the others.

Figures 5.5 and 5.6 show the two video frames and their ground-truth, and the separation obtained by all algorithms.

We conclude by reminding the reader that only two ground-truth frames were available for this excerpt of the video, and that only 20 are available for the full 3584 video frames. We recommend performing more experiments on similar benchmarks before drawing definitive conclusions

Algorithm	**AUC**	**Parameter 1**	**Parameter 2**	**Iterations (indicative)**	**Time (indicative)**
RPCA2D GL ℓ_1	0.8945	0.000595	1	46	212.08
RPCA2D Fro ℓ_1	0.8841	0.000414		30	137.22
RPCA2D Fro ℓ_2	0.883	0.000595		32	149.48
Tensor RPCA (CPVR2016)	0.8645	0.00087		205	673.38
NC TRPCA	0.8633	48	1000	99	506.23
BRTF	0.8616	0.0621	100	111	98.41
Tensor HORPCA-S (Goldfarb & Qin)	0.8603	0.0128		102	461.07
Tensor RHOSVD	0.858	0.00316	1	67	538.48
Tensor BRPCA	0.8577	0.01	0.6	84	101.85
RPCA2D GL ℓ_2	0.8559	0.000288	1	32	220.55
Tensor RCPD (sub)	0.8558	0.000316	50	77	110.05
Tensor RPCA	0.8554	0.00316		72	314.74
Tensor RCPD (lin)	0.8322	0.01	20	72	51.69
Tensor RPCA (CPVR2014)	0.7558	0.00754		200	659.86
Cauchy Soft-Thresholding	0.7558	5.84	0.05	501	676.34
Welsh Soft-Thresholding	0.7089	10.9	0.236	501	484.82

Table 5.5: Classification performance on the *Airport Hall* dataset in descending order of AUC

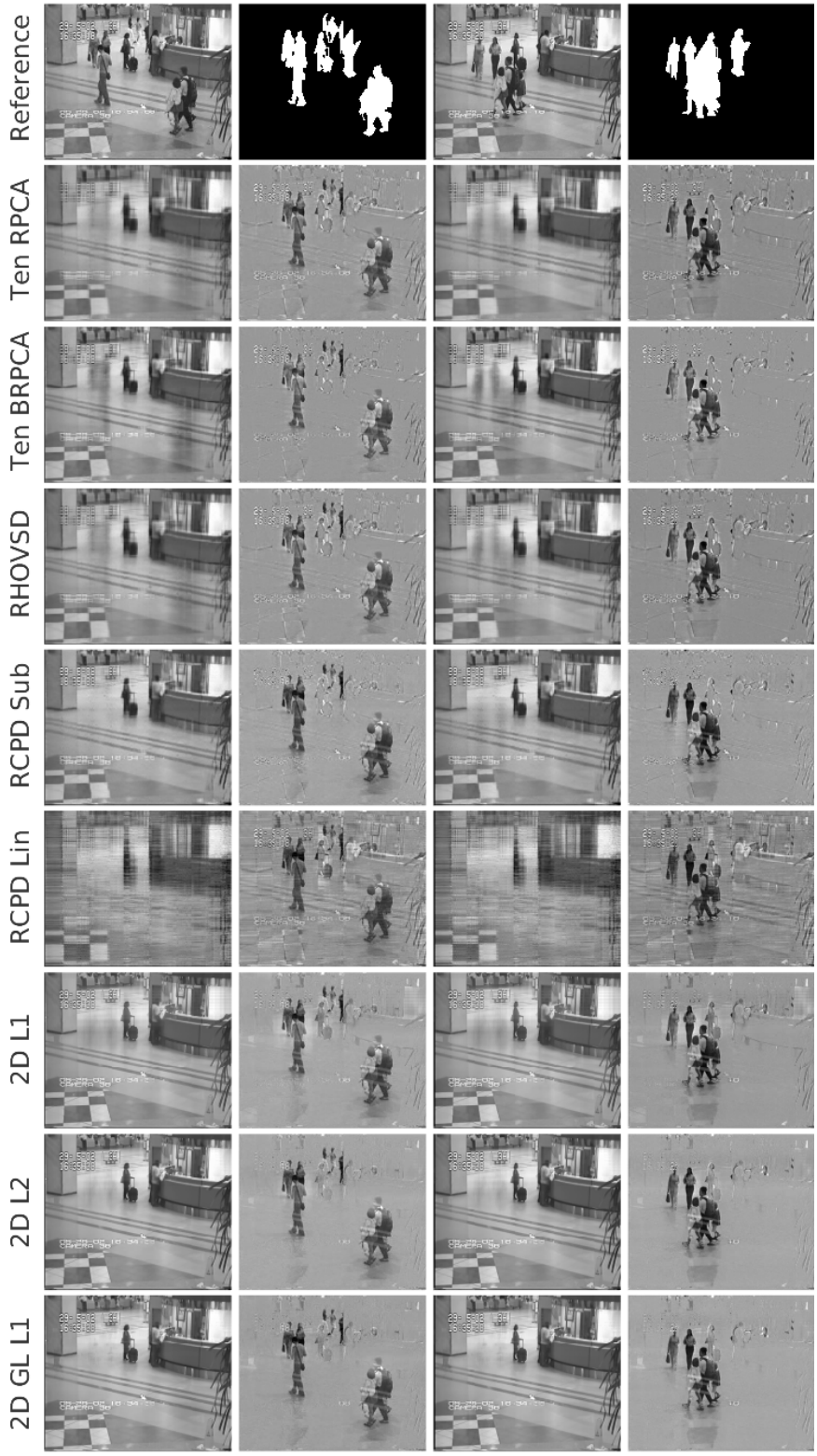

Figure 5.5: Background subtraction results on the *Airport Hall* dataset - part 1

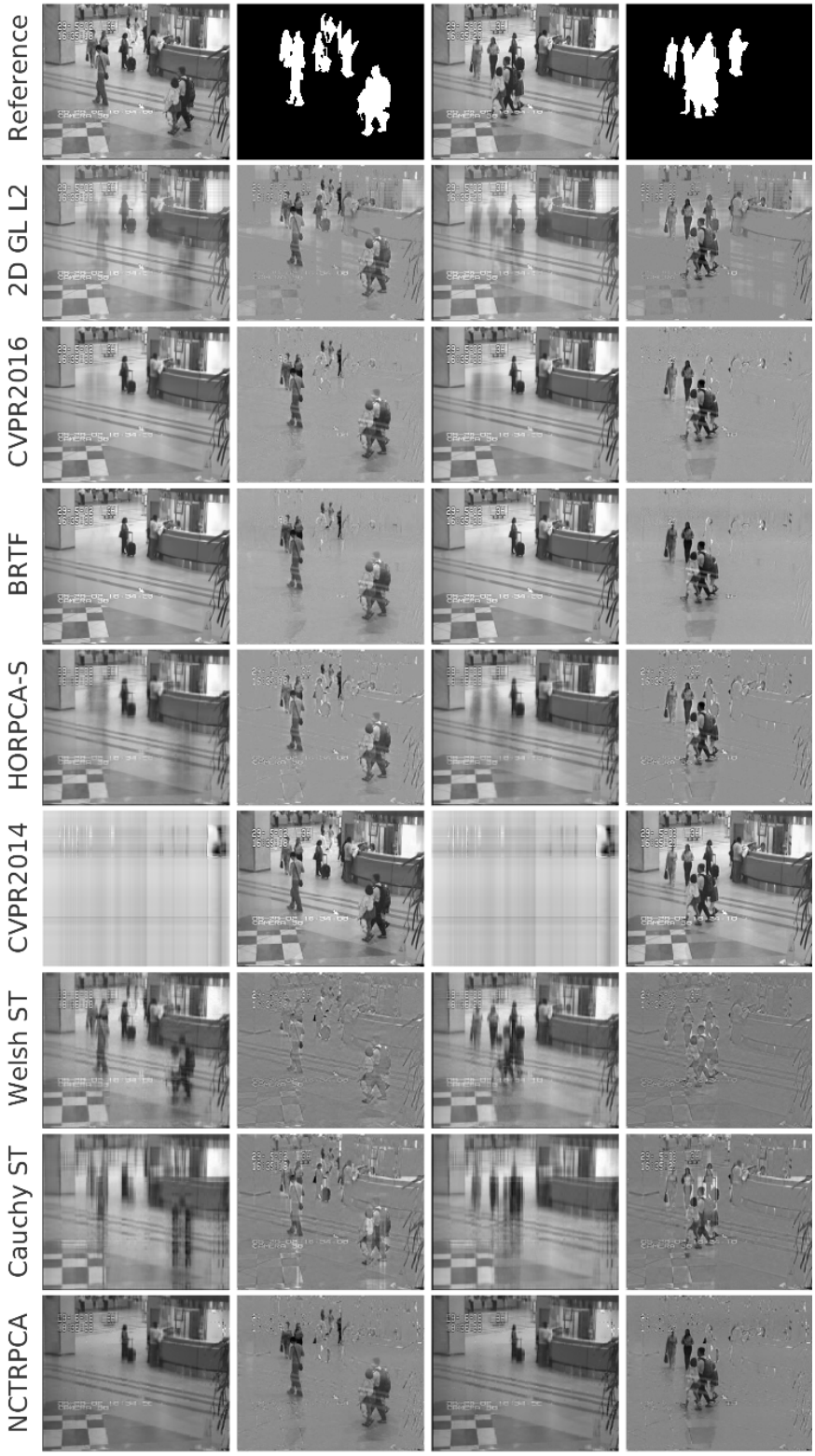

Figure 5.6: Background subtraction results on the *Airport Hall* dataset - part 2

5.3 Denoising of monochromatic face images

One of the most well-known applications of Robust PCA methods is image denoising. In this section we present the results obtained by comparing the performance of the 15 methods of the panel on a subset of the Extended Yale-B dataset. This dataset comprises face images of 10 different subject, each under 64 different lighting conditions. According to Ramamoorthi and Hanrahan [80], Basri and Jacobs [10], face images of one subject under various lighting lie approximately on a 9-dimensional subspace, and are therefore suitable candidates for low-rank modelling.

We use the pre-cropped images and keep them at full resolution and experiment on the set of face photographs of the first subject. This gives us a 3-way tensor of 64 images of size 192×168 where the three mode correspond respectively to the columns and rows of the image, and to the illumination component. We evaluate the recovery performance at increasing levels of corruption by sparse Laplacian noise, as well as the reconstruction accuracy on images with partial obstructions by rectangular patches of bounded random size and of random content.

All three modes are expected to be low-rank due to the spatial correlation within frontal slices and to the correlation between images of different illumination. In this experiment, we do not set bounds on the ranks manually and we optimise the parameters given in Table 5.1.

5.3.1 Salt & pepper noise

Salt & Pepper noise is a commonly encountered type of noise, that manifests itself as random sparse corruption by black and white pixels, and suits the Laplacian noise hypothesis. We test three different corruption levels: 10%, 30%, and an extreme case of 60% corruption.

The range of values tested for each algorithm's parameters is that of Table 5.2. The only exception is that we set $\alpha_t = 1e-3$ for our 2D methods when the noise level is below 30%, and $\alpha_t = 1e-2$ otherwise. We measure the recovery accuracy according to four different metrics: PSNR, SSIM, FSIM and the mean value of the SAM (MSAM for short). Although the images are not multispectral, we use MSAM to measure how well the luminance bands are preserved. It is interesting to see the black noise pixels as missing values and to test a variant of *HORPCA - HORPCA-TC* - built to handle incomplete data. *HORPCA-TC* demonstrates increased accuracy at low noise levels but performs worse than then standard *HORCPA* on gross corruption.

Our measure of choice for determining which images to compare visually is the FSIM for its higher correlation to subjective evaluation than PSNR and SSIM. Since the tensor we observe is a collection of images, the four quantitative metrics we report are averages over the 64 frontal slices.

We present a summary of the quantitative performance measures of the panel in Table 5.6. At the 10% noise level, nearly every method provides good to excellent recovery of the original images and the differences between the best methods are in the preservation of fine details such as the reflection of the light in the subject's eyes, as can be seen in Figure 5.7. This detail in particular is very close in nature to the white pixel corruption of the salt & pepper noise, so it is quite impressive that *RPCA2D Fro* ℓ_1 and *RPCA2D GL* ℓ_1 are able to preserve it. *RPCA2D GL* ℓ_2 also preserves the sparkles but doesn't remove all of the noise, so its performance is overall disappointing. Figure 5.8 presents a detail of the third image of the dataset to allow for close inspection and illustrate the effects of shadows on recovery.

The image reconstructed by *BRTF* is very close to the mean and the specific features of the image have all been removed; this explains the low FSIM and PSNR. *Tensor RPCA from CVPR2014* is visually the worst of all reconstructions and the fact that the algorithm has a higher PSNR than *BRTF* goes to show that PSNR alone is not sufficient for accurate comparisons.

The methods from [76] are competitive with the exception of *RCPD*, we ought to highlight the high similarity between the scores *HORPCA-S* and *Tensor RPCA* on all metrics, which was expected

Algorithm	Best PSNR	Best SSIM	Best FSIM	Best MSAM
RPCA2D GL ℓ_1 (Bahri et. al.)	36.4236	0.9619	0.9867	0.0312
RPCA2D Fro ℓ_1 (Bahri et. al.)	34.1806	0.9363	0.9835	0.0442
Cauchy ST	36.8013	0.9526	0.9826	0.0374
RPCA2D Fro ℓ_2 (Bahri et. al.)	36.374	0.9717	0.9786	0.0379
Tensor RPCA (CVPR 2016)	35.145	0.9309	0.9779	0.0471
HORPCA-S TC	35.0652	0.9456	0.9771	0.0485
Tensor BRPCA (Papamakarios et. al.)	35.0819	0.9591	0.9728	0.0452
HORPCA-S	34.3085	0.9565	0.9698	0.0498
Tensor RHOSVD (Papamakarios et. al.)	33.6492	0.9577	0.9675	0.0556
Tensor RPCA (Papamakarios et. al.)	34.066	0.9585	0.9661	0.0525
Welsh ST	31.6702	0.8756	0.9539	0.0698
NC TRPCA	23.6362	0.6684	0.8839	0.1941
Tensor RCPD (Papamakarios et. al.)	24.0525	0.6983	0.8834	0.1782
RPCA2D GL ℓ_2 (Bahri et. al.)	25.2376	0.619	0.8713	0.1391
BRTF	16.7486	0.6048	0.8241	0.3703
Tensor RPCA (CVPR 2014)	19.1429	0.4696	0.7902	0.2757

Table 5.6: Best performance on Yale-B sorted by FSIM - 10% Salt & Pepper noise

since both models are similar in design. The bilinear version of *Tensor RPCA* performs slightly better than the standard algorithm and exhibits a higher PSNR than *RPCA2D Fro* ℓ_1, but a lower FSIM and higher MSAM. Visual inspection of the results confirm a higher reconstruction quality for the 2D model and show that the performance of *Tensor BRPCA* is closer to that of *RPCA2D Fro* ℓ_2. Out of the two robust factorisations, *RHOSVD* has better performance than *RCPD*, but is not among the best models on this benchmark. Out of the two *redescending M-estimator* algorithms, *Cauchy ST* has the upper hand and offers excellent recovery while *Welsh ST* failed to remove all the noise from the image. Finally, the non-convex *NC TRPCA* effectively removes the salt and pepper noise but introduces artifacts of its own. The recovered low-rank image appears choppy and details of the skin and of the eyes are lost in the process.

Clearer differences appear at the 30% noise level, as demonstrated both by the quantitative measures presented in Table 5.7, and by visual inspection of Figures 5.9 and 5.10. Overall, the recovery performance of the models is markedly lower and most method start to lose much of the features of the face and regress towards the mean - similarly to *BRTF*. The four methods from Papamakarios [76] and the two models from Goldfarb and Qin [40] again give very similar results, both subjectively and quantitatively. We observe little variations in the performance of *NC TRPCA*, which suggests that the method is robust to noise even though the reconstruction performance we obtained is not as high as other models. The same can be said about *BRTF*, which gave a very similar result to the 10% noise case. *Tensor RPCA from CVPR2014* and *RPCA2D GL* ℓ_2 gave the worst two results, while this was to be expected for the *CVPR2014* model given its results on the 10% noise benchmark, the performance of the 2D model plummeted and is not comparable to that of the other three 2D algorithms anymore.

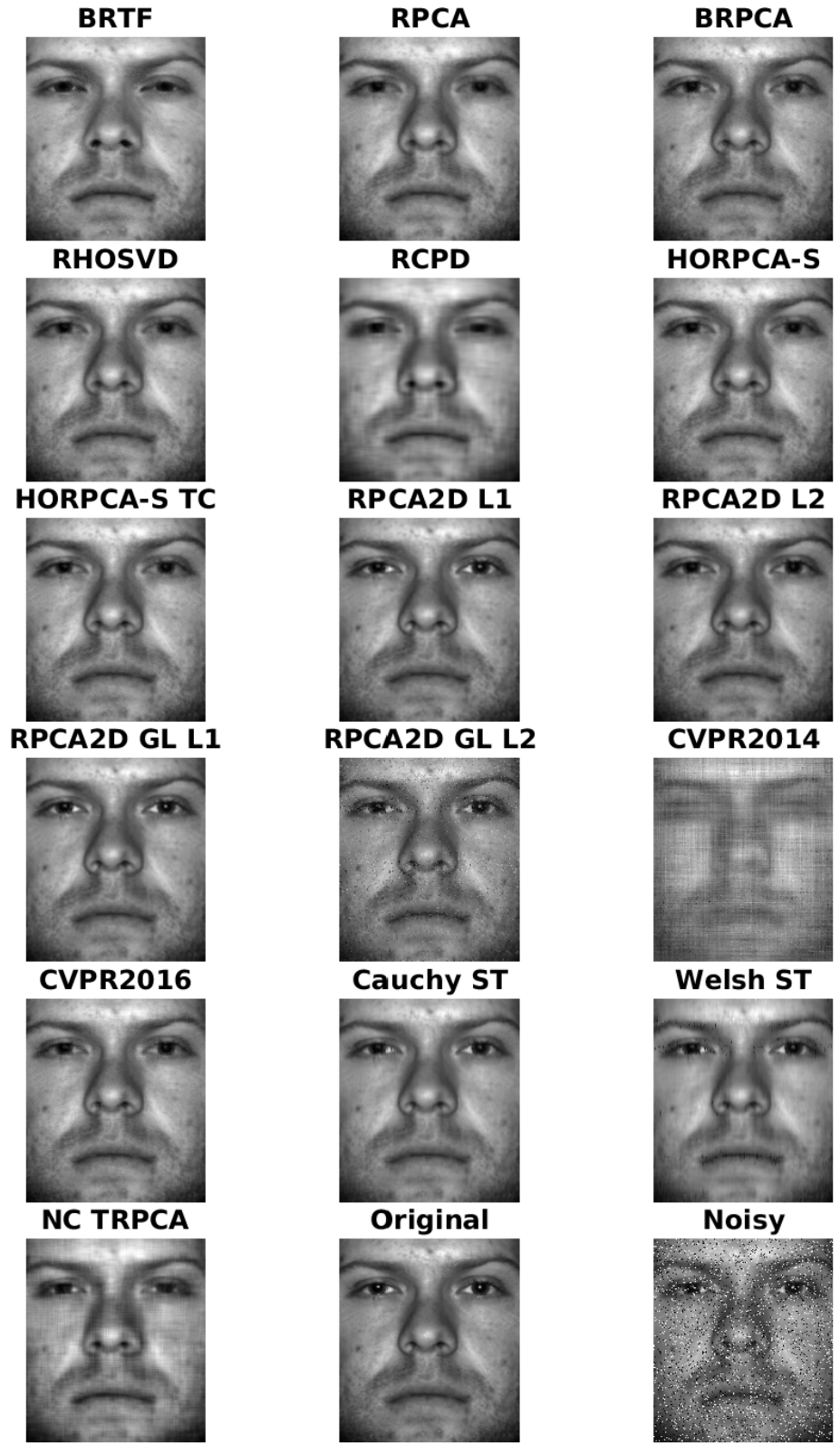

Figure 5.7: Best results per FSIM on Yale-B with 10% S & P noise - first image

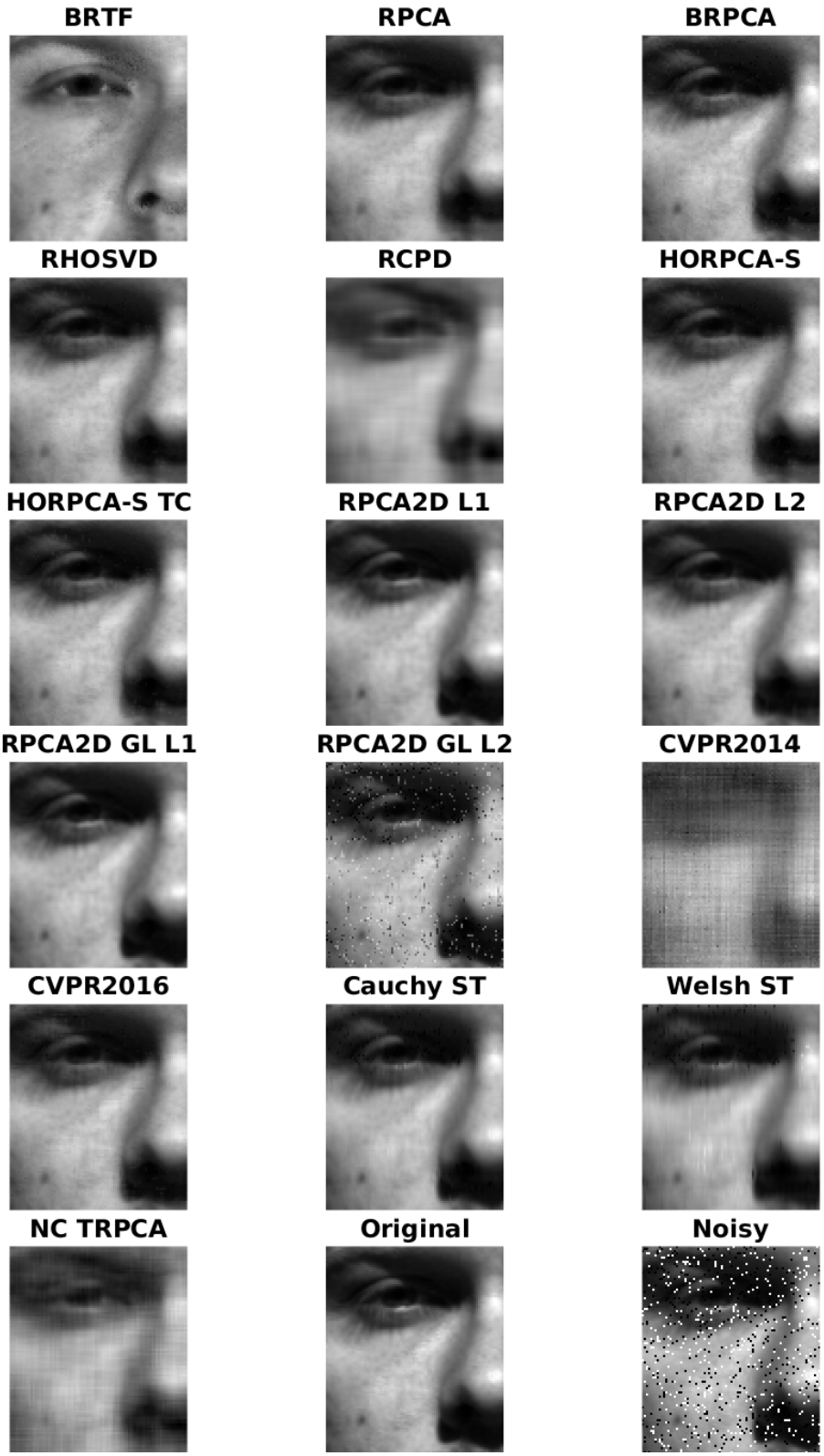

Figure 5.8: Best results per FSIM on Yale-B with 10% S & P noise - detail of the third image

The best performing methods, both visually and quantitatively are again *RPCA2D Fro* ℓ_1, *RPCA2D GL* ℓ_1, *RPCA2D Fro* ℓ_2, *Cauchy ST*, and the *CVPR2016* model although noticeably less smooth and noisier than the previous four. Compared to the 10% noise case, the performance of our 2D model with ℓ_1-regularisation on the core tensor are similar; fine details of the face are still well preserved and we can still perceive the light reflection in the eyes. Note however that the result produced by *RPCA2D GL* ℓ_1 is more rugged than that of *RPCA2D Fro* ℓ_1.

Algorithm	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
RPCA2D Fro ℓ_1 (Bahri et. al.)	31.9094	0.881	0.9568	0.0660
RPCA2D Fro ℓ_2 (Bahri et. al.)	31.1351	0.9234	0.9474	0.0646
RPCA2D GL ℓ_1 (Bahri et. al.)	30.8656	0.8628	0.947	0.0790
Cauchy ST	30.7079	0.8752	0.9424	0.0729
HORPCA-S TC	30.0806	0.8988	0.9352	0.0836
Tensor RPCA (CVPR 2016)	28.6409	0.8023	0.9299	0.1030
HORPCA-S	28.6301	0.8837	0.9179	0.0965
Tensor RPCA (Papamakarios et. al.)	28.5048	0.8468	0.9103	0.0994
Tensor BRPCA (Papamakarios et. al.)	28.9468	0.8809	0.9096	0.0943
Tensor RHOSVD (Papamakarios et. al.)	26.9728	0.8741	0.9033	0.1182
Welsh ST	25.8976	0.7029	0.9008	0.1296
Tensor RCPD (Papamakarios et. al.)	24.4019	0.718	0.8841	0.1645
NC TRPCA	22.5348	0.6563	0.8816	0.2192
BRTF	17.6424	0.5394	0.8176	0.3682
Tensor RPCA (CVPR 2014)	17.1359	0.3931	0.7627	0.3366
RPCA2D GL ℓ_2 (Bahri et. al.)	16.6938	0.3631	0.7558	0.3772

Table 5.7: Best performance on Yale-B sorted by FSIM - 30% Salt & Pepper noise

Finally, we present in Table 5.8 and in Figures 5.11 and 5.12 the recovery results in the extreme case of 60% corruption. The top five algorithms according to FSIM are, in order, *RPCA2D Fro* ℓ_1, *Tensor RCPD*, *RPCA2D Fro* ℓ_2, *NC TRPCA*, and *CVPR2016*. Subjective assessment of the visual quality confirm that these methods are the five best, but suggests that the image recovered by *NC TRPCA* is closer to the original image than the one produced by *RPCA2D Fro* ℓ_2. Interestingly, *RCPD* has a higher PSNR than in both previous benchmarks and a nearly identical FSIM.

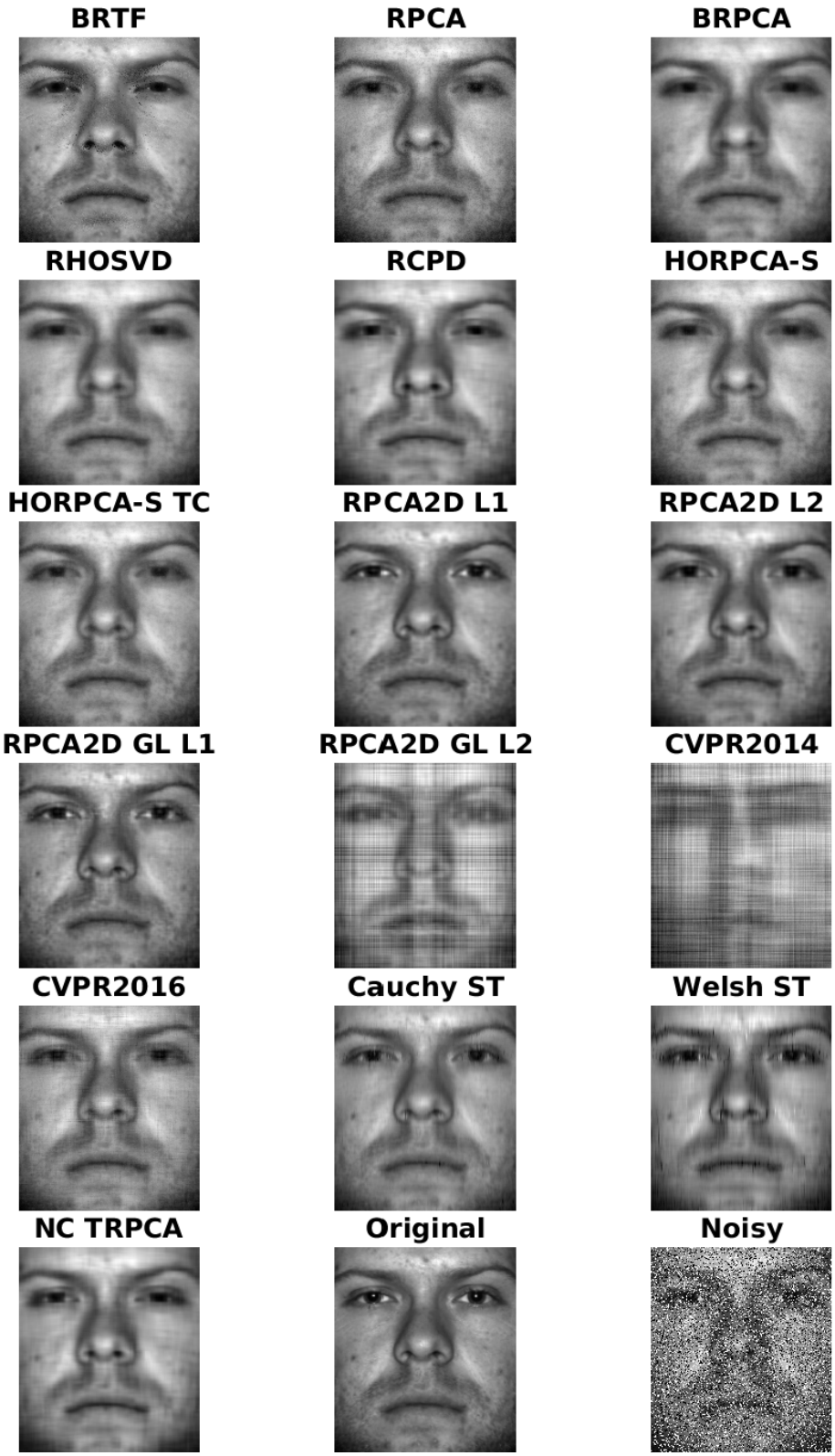

Figure 5.9: Best results per FSIM on Yale-B with 30% S & P noise

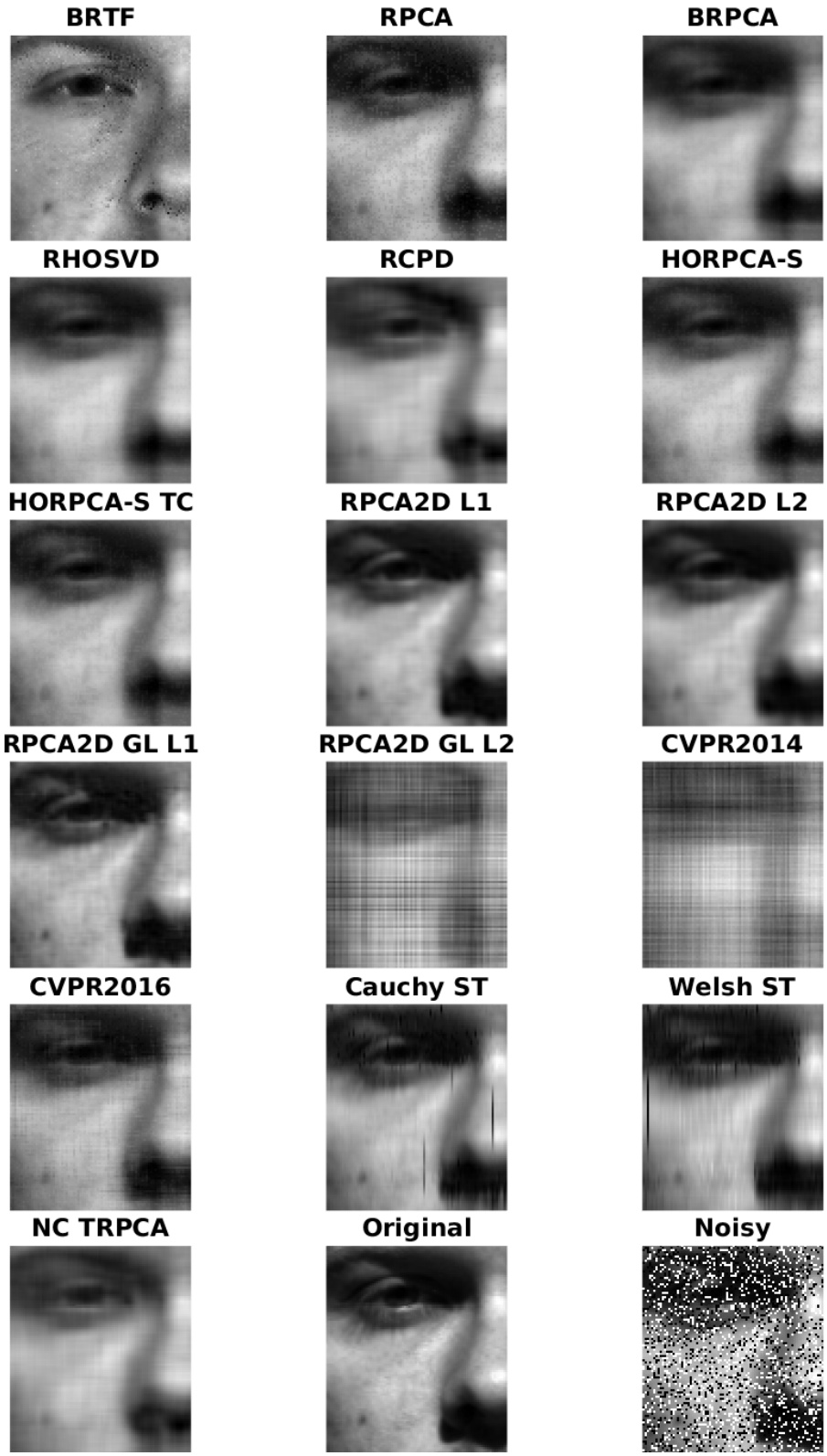

Figure 5.10: Best results per FSIM on Yale-B with 30% S & P noise - detail of the third image

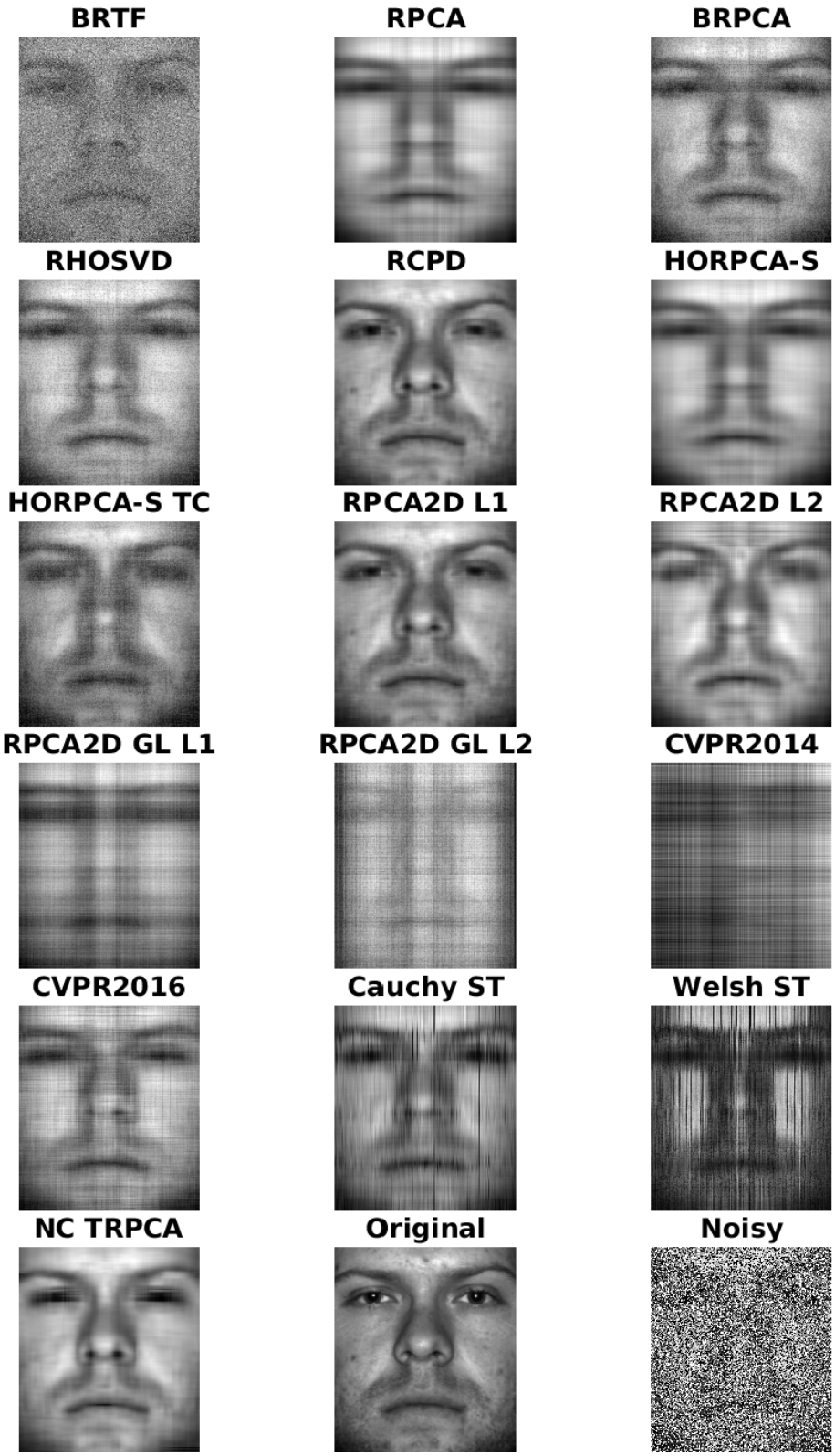

Figure 5.11: Best results per FSIM on Yale-B with 60% S & P noise

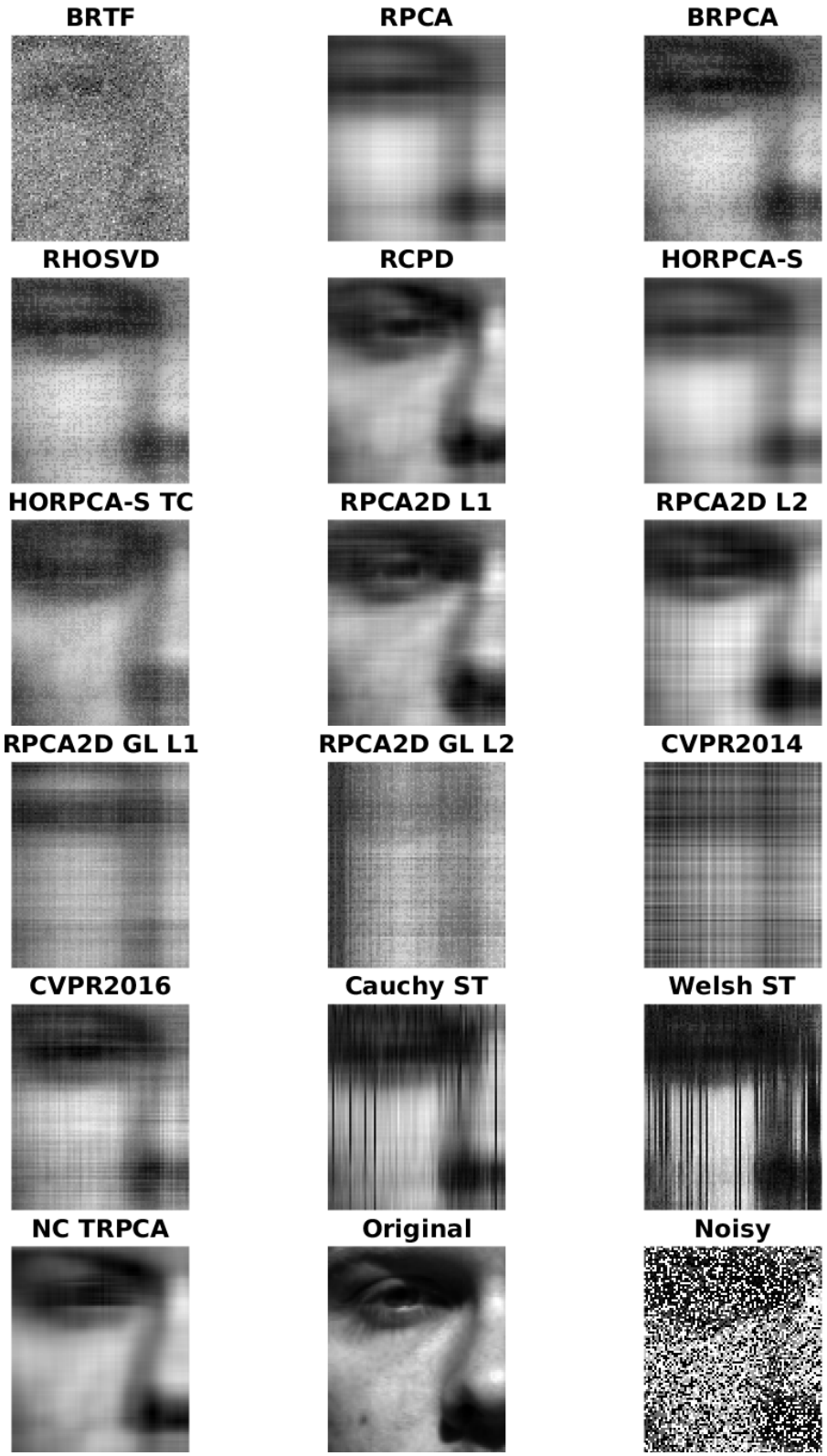

Figure 5.12: Best results per FSIM on Yale-B with 60% S & P noise - detail of the third image

Algorithm	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
RPCA2D Fro ℓ_1 (Bahri et. al.)	26.6057	0.7392	0.8956	0.1286
Tensor RCPD (Papamakarios et. al.)	25.2225	0.717	0.8823	0.1568
RPCA2D Fro ℓ_2 (Bahri et. al.)	21.3406	0.696	0.8648	0.1688
NC TRPCA	22.8502	0.6535	0.8509	0.2091
Tensor RPCA (CVPR 2016)	22.566	0.5926	0.8427	0.2046
Cauchy ST	20.8269	0.5292	0.8131	0.2243
HORPCA-S	22.141	0.6607	0.8115	0.2030
HORPCA-S TC	20.8387	0.5582	0.8062	0.2124
Tensor RPCA (Papamakarios et. al.)	19.8659	0.6286	0.8055	0.2521
Tensor BRPCA (Papamakarios et. al.)	20.7134	0.5894	0.7964	0.2203
Tensor RHOSVD (Papamakarios et. al.)	20.277	0.5655	0.7857	0.2400
RPCA2D GL ℓ_1 (Bahri et. al.)	14.1467	0.3191	0.7461	0.4848
Welsh ST	16.5715	0.4368	0.7335	0.3624
Tensor RPCA (CVPR 2014)	15.0052	0.2872	0.7201	0.4274
RPCA2D GL ℓ_2 (Bahri et. al.)	12.7937	0.2042	0.6787	0.5452
BRTF	13.0653	0.132	0.5849	0.5219

Table 5.8: Best performance on Yale-B sorted by FSIM - 60% Salt & Pepper noise

5.3.2 Random patch occlusion

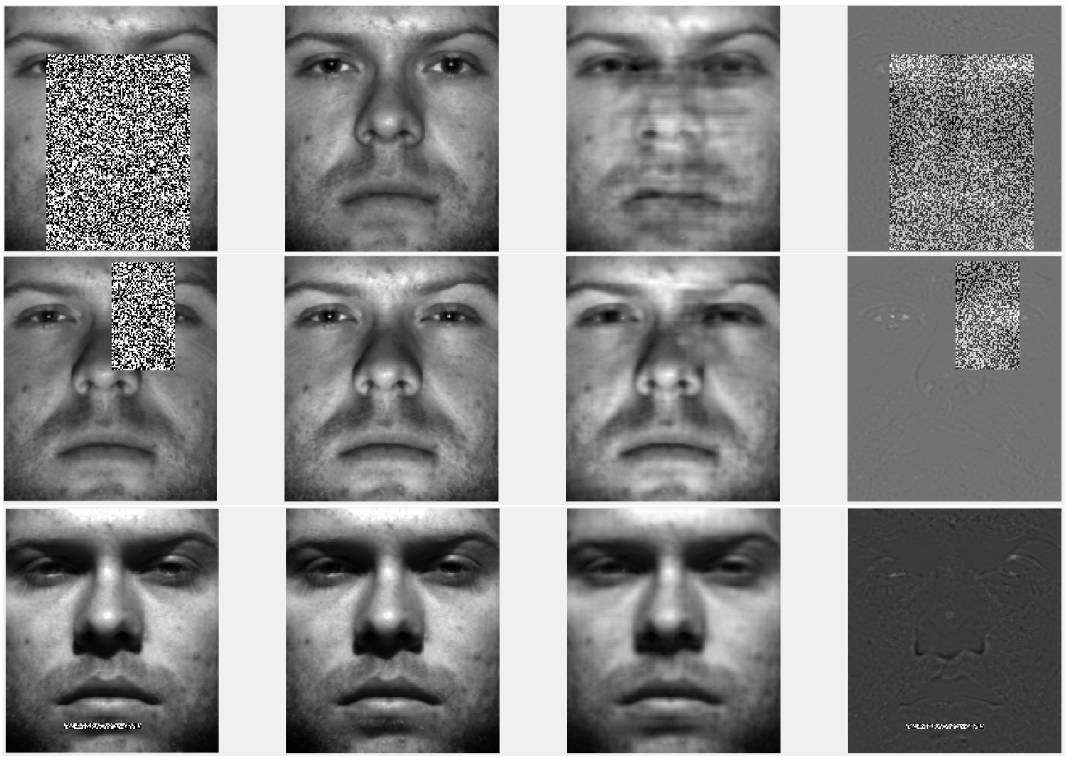

Figure 5.13: Sample results for *RPCA2D Fro* ℓ_2 with robust estimation of the mean on three different patch sizes for a maximum patch size of 160 pixels

We generate random patches of variable maximum size and of random non-Gaussian content; the corruption is therefore still sparse but is now structured. We test three different patch sizes for corrupting the 192×168 images of the first subject of the cropped Yale-B database: 50, 100, and 160 pixels.

In this experiment, we also evaluate the effectiveness of the robust estimator of the mean embedded in our algorithms. We provide an example recovery of the noise-free mean in Figure 5.14 and report the performance of our four methods both with and without the estimator. As for the salt and pepper corruption, we rely on the FSIM index to determine which recovered image we present as the best for each algorithm and the measures reported are averages over all the frontal slices.

As can be seen from Figure 5.14, the estimator provides a close reconstruction of the noise-free mean. Quantitative evaluation of the reconstruction quality is summarised in Table 5.9 and confirms that the recovered mean is close to the true mean, structurally and in the ℓ_2 sense.

Metric	PSNR	FSIM	SSIM	SAM	Relative error
Value	31.1572	0.9758	0.9638	8.93e-05	0.0395

Table 5.9: Quantitative assessment of the reconstructed mean for a maximum patch size of 160px on a sample run

Table 5.10 and Figures 5.15 and 5.16 respectively present the quantitative quality metrics and the best recovery by FSIM and by PSNR for each model. At such a small patch size, some methods

Figure 5.14: True mean (left) and recovered mean (right) by *RPCA2D Fro* ℓ_2 with robust estimation of the mean for a maximum patch size of 160 pixels

already fail to remove or reduce the corruption: *RPCA2D GL* ℓ_2 both with and without estimation of the mean, *Cauchy ST*, *Welsh ST*, and *Tensor RPCA (CVPR2014)* all show little to no reduction of the corruption. *NC TRPCA* provides some recovery of the corrupt area but at the same time degrades the quality of the rest of the image. To a certain extent, *RPCA2D Fro* ℓ_1 exhibits a somewhat similar behaviour as we can see that some parts of the picture become blurred, especially without the estimation of the mean. Interestingly, *RPCA2D GL* ℓ_1 provides a better reconstruction than its Frobenius-penalised counterpart. *BRTF* behaves identically to the salt & pepper noise benchmarks and removes both the corruption and the specific features of the image.

On this benchmark, mode-n matricisation-based algorithms seem to perform the best. *Tensor RPCA* from [76] and HORPCA-S [40] are similar models and have nearly identical performance. *Tensor BRPCA* from [76] is the best according to all metrics and visually. *RHOSVD* performs noticeably better than *RCPD* and provides a reconstruction visually similar to that of *RPCA*, *HORPCA*, and *BRPCA*. Out of our 2D models, *RPCA2D Fro* ℓ_2 is by far the best and ranks 5^{th} according to FSIM and PSNR. The other 2D models do not seem to be as competitive with the best algorithms but provide acceptable recovery with the exception of *RPCA2D GL* ℓ_2 which again proves unable to denoise the image.

We ought to question the suitability of FSIM on this confined corruption benchmark based on visual inspection of the result. *Tensor RPCA (CVPR2016)* has a **lower** FSIM score than *Cauchy ST*, but visual comparison of the results clearly shows that there is less remaining corruption in the image recovered by *Tensor RPCA (CVPR2016)*. Similarly, comparing the best image selected by FSIM and PSNR shows that the best image by PSNR is of equal or higher visual quality than the best image by FSIM. This leads us to choose PSNR as a reference measure for this benchmark, and, as a consequence, to order the results tables by PSNR.

Similarly to the 50px patch size case, we present in Tables 5.11 and 5.12, and in Figures 5.17, 5.18, 5.19, and 5.20 the denoising results for patches of maximum size 100px and 160px. The first observation we can make is that the top 5 methods are the same as in the 50px case, with the difference that *RPCA2D Fro* ℓ_2 with the estimator of the mean now performs better than the variant that doesn't estimate the sample mean. Otherwise, the best methods are the same and visually appealing results are obtained in the 100px case. In the 160px patch case, the patch can obstruct the majority of the image and all methods demonstrate a noticeable decrease in performance.

We again have to question the suitability of quantitative measures for this benchmark. While PSNR seems to be a better indicator of the quality of the reconstructed image in general, we have to report some exceptions that prove this choice is not optimal either. Comparing the best images

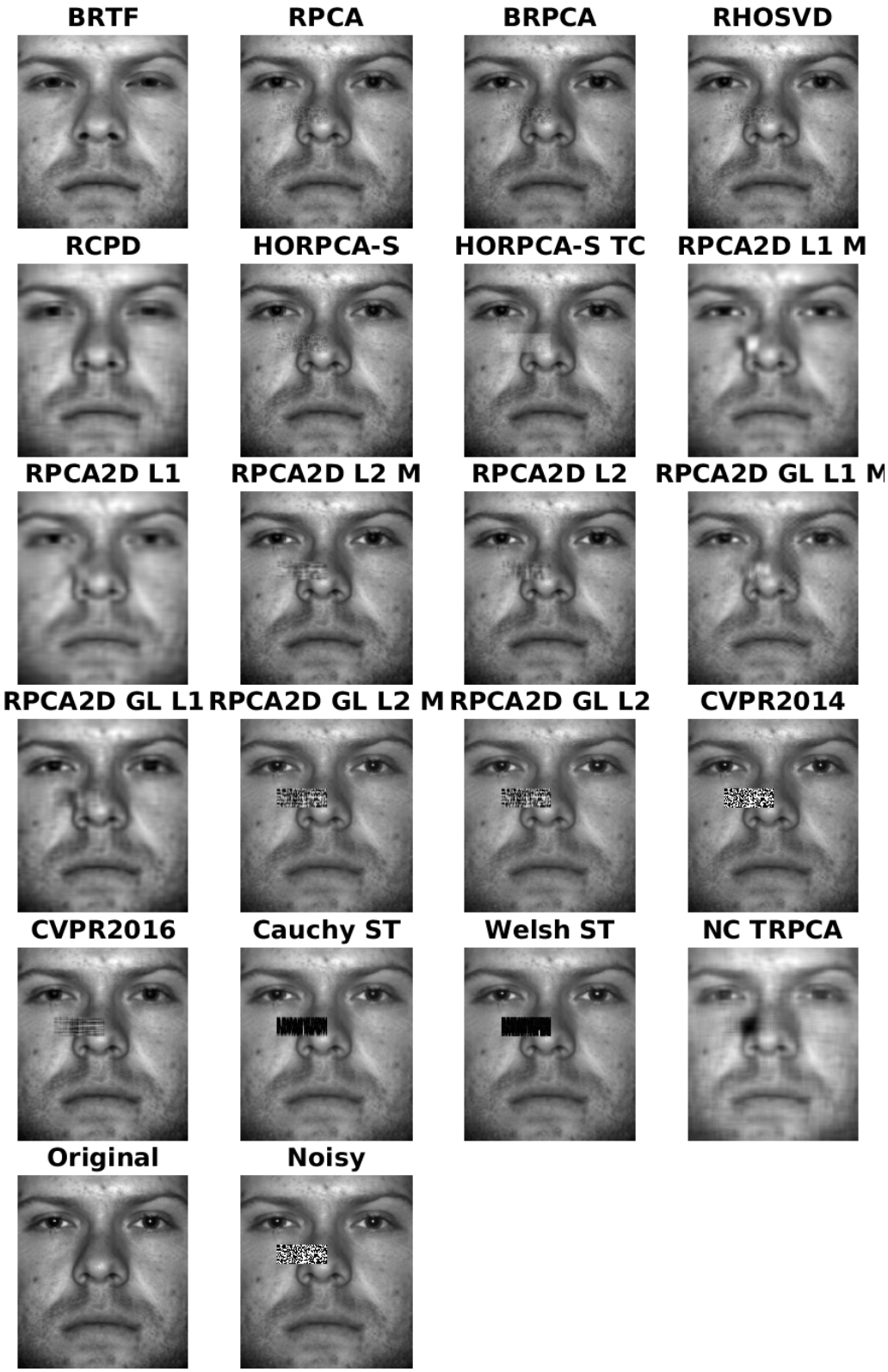

Figure 5.15: Best results per FSIM on Yale-B with patch of max size 50px

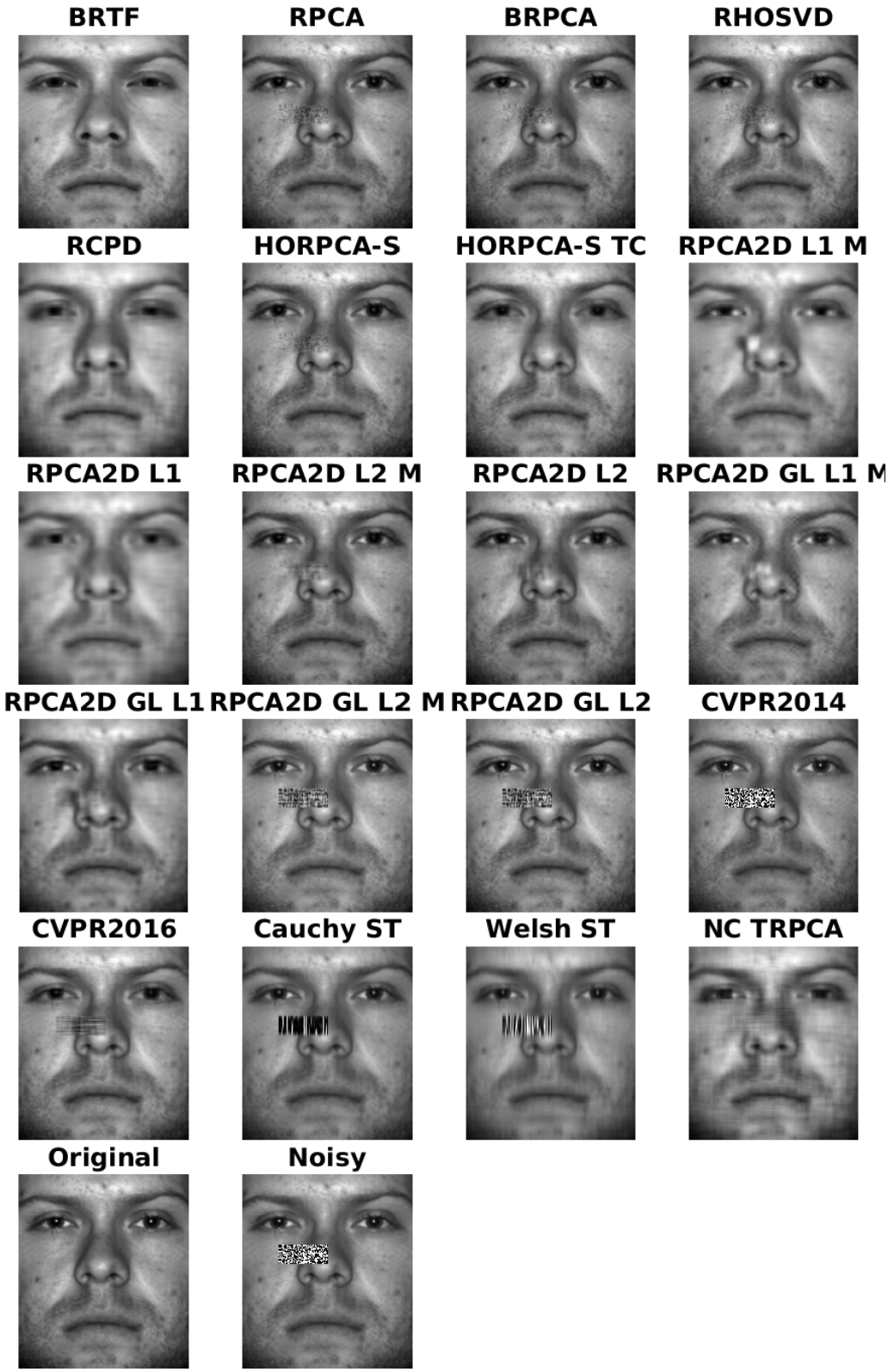

Figure 5.16: Best results per PSNR on Yale-B with patch of max size 50px

Algorithm	Best PSNR	Best SSIM	Best FSIM	Best MSAM
Tensor BRPCA (Papamakarios et. al.)	38.6465	0.9836	0.9887	0.0286
Tensor RPCA (Papamakarios et. al.)	37.585	0.981	0.9872	0.0324
HORPCA-S	37.3662	0.9792	0.987	0.0337
Tensor RHOSVD (Papamakarios et. al.)	37.264	0.9806	0.9867	0.0335
RPCA2D Fro ℓ_2 (Bahri et. al.)	33.7598	0.9478	0.9833	0.0554
RPCA2D Fro ℓ_2 w/ mean (Bahri et. al.)	33.2748	0.9269	0.9823	0.0597
Tensor RPCA (CVPR 2016)	32.5854	0.9216	0.9811	0.0642
HORPCA-S TC	30.1705	0.9081	0.9798	0.0849
Cauchy ST	28.4654	0.9585	0.982	0.1084
RPCA2D Fro ℓ_1 (Bahri et. al.)	27.0376	0.7571	0.9077	0.1221
RPCA2D GL ℓ_2 (Bahri et. al.)	27.0125	0.8964	0.9629	0.1176
RPCA2D GL ℓ_2 w/ mean (Bahri et. al.)	26.3324	0.8673	0.9629	0.1260
RPCA2D Fro ℓ_1 w/ mean (Bahri et. al.)	25.3586	0.7554	0.9124	0.1569
Welsh ST	25.2723	0.9585	0.9721	0.1412
RPCA2D GL ℓ_1 (Bahri et. al.)	23.5498	0.7285	0.9199	0.1935
Tensor RCPD (Papamakarios et. al.)	23.5418	0.6885	0.8815	0.1867
Tensor RPCA (CVPR 2014)	22.0006	0.9585	0.965	0.1942
NC TRPCA	21.5563	0.6427	0.8807	0.2334
RPCA2D GL ℓ_1 w/ mean (Bahri et. al.)	19.5462	0.6626	0.9071	0.2710
BRTF	16.6836	0.6175	0.8271	0.3659

Table 5.10: Best performance on Yale-B sorted by PSNR - Patch of max size 50px

by PSNR and FSIM for *NC TRPCA*, *RPCA2D GL* ℓ_1 and shows that lesser quality images can wrongly be selected (as is the case for *NC TRPCA* in the 100px case), and that when the algorithm removes little of the corruption, it is difficult to choose which quantitative metric is best. Let us consider the output of *RPCA2D GL* ℓ_1 without mean estimation in the 100px case: in the image selected by FSIM, the patch is virtually unaltered and so is the rest of the image. Therefore, all but the corrupt area is of high quality but the method fails to remove the noise. On the other hand, the image selected by PSNR is very different from the original image and of low visual quality, but the patch has been removed. Based on these observations, we conclude that automated selection of the highest quality image is difficult and that the choice of the quantitative metric is problem-dependent.

Conclusion

The 2D methods that use ℓ_2 regularisation on the bases are able to reduce patch corruption, and visual and quantitative inspection show that the robust mean estimation procedure can have both a positive and a negative impact on the quality of the reconstructed image. We believe more experiments are needed to accurately quantify the effect of this procedure on patch denoising performance.

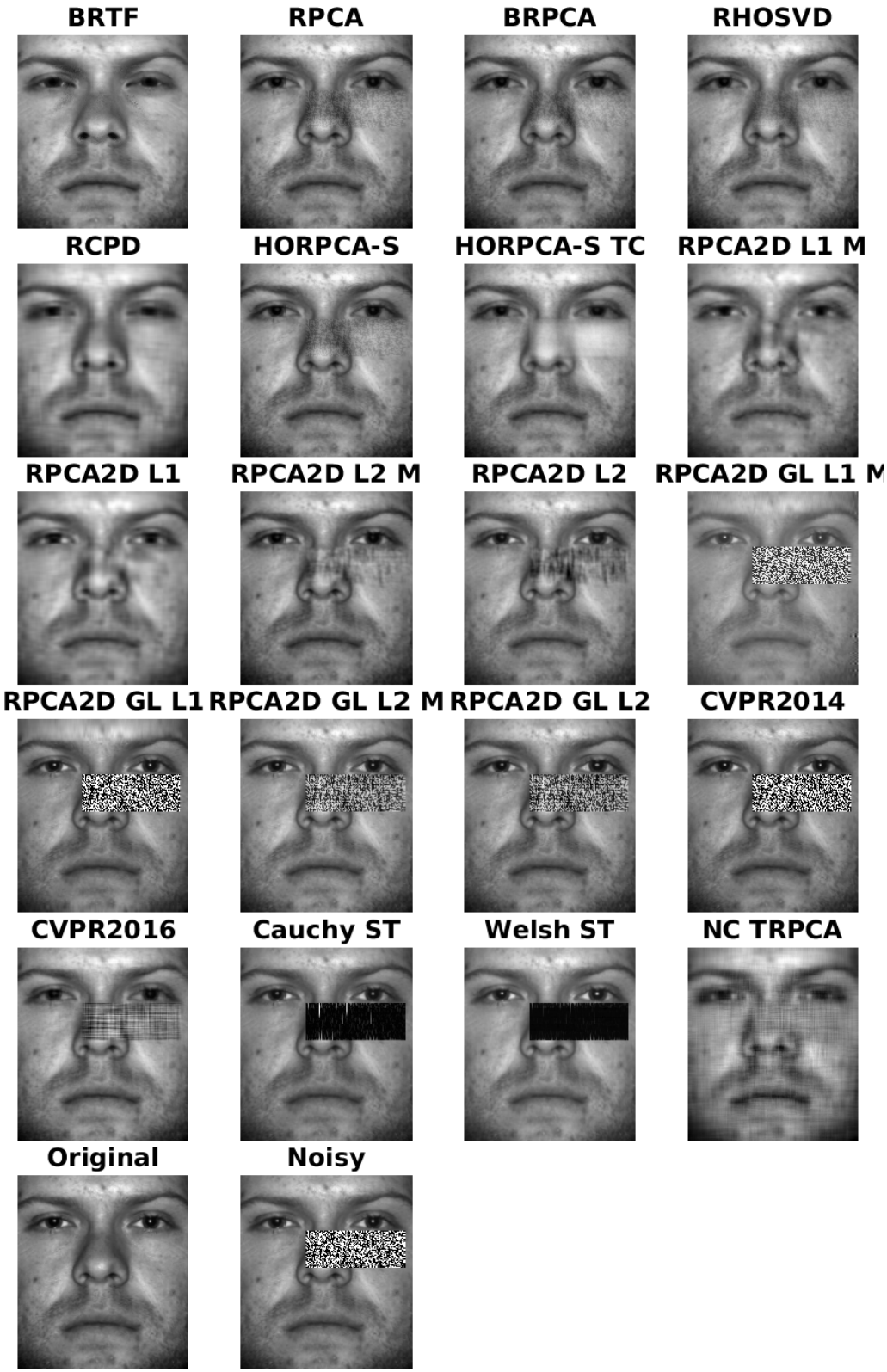

Figure 5.17: Best results per FSIM on Yale-B with patch of max size 100px

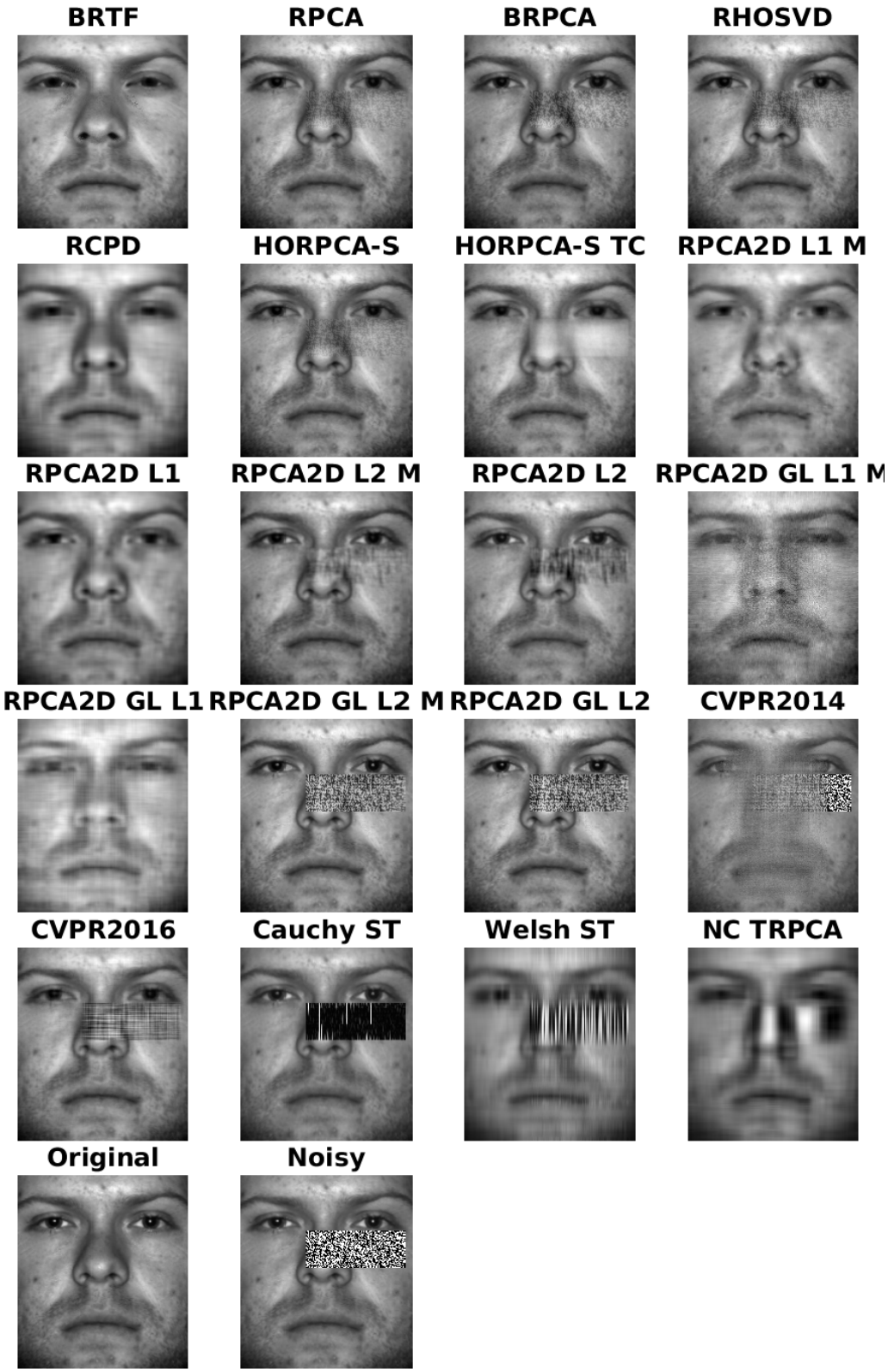

Figure 5.18: Best results per PSNR on Yale-B with patch of max size 100px

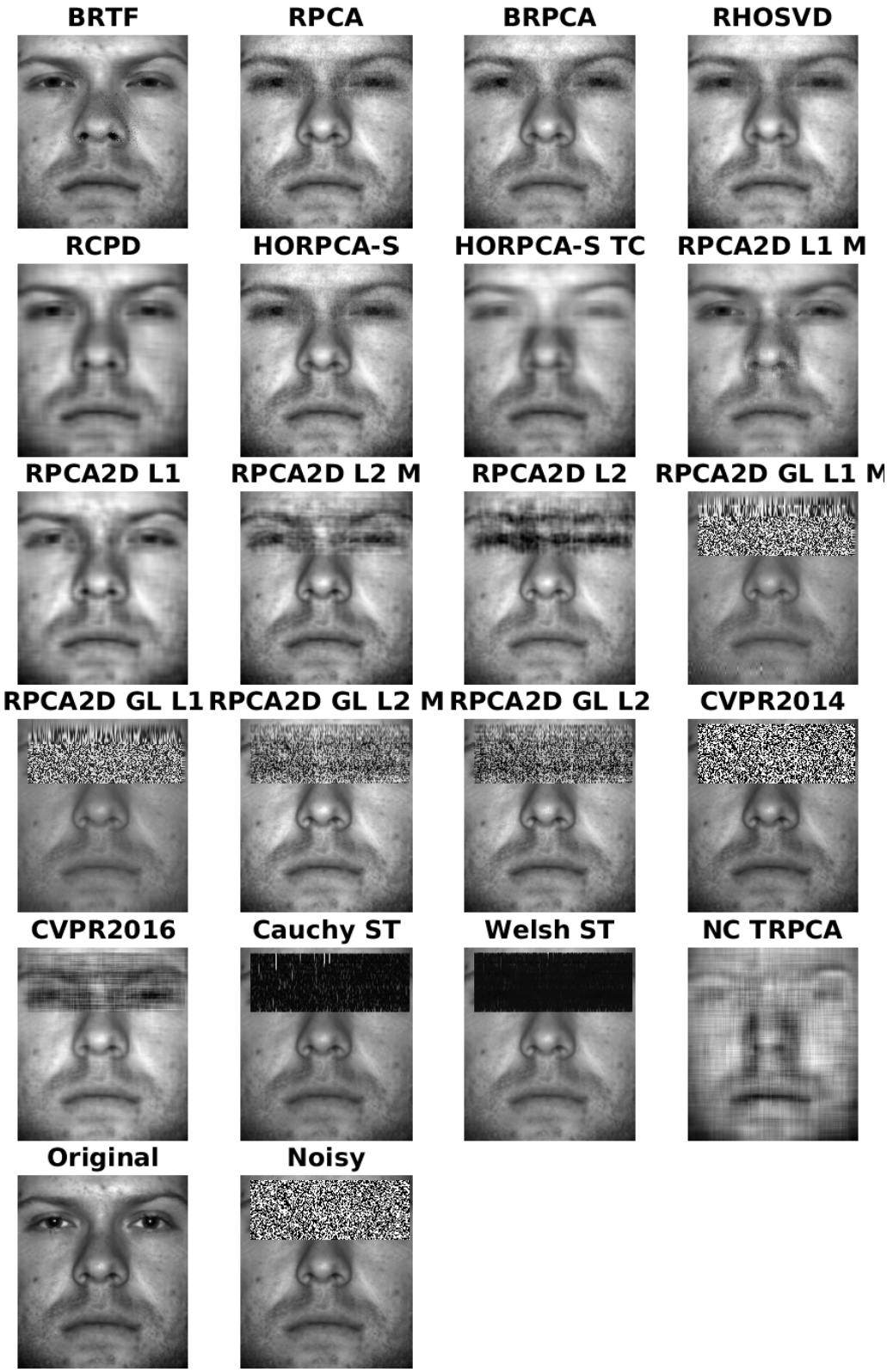

Figure 5.19: Best results per FSIM on Yale-B with patch of max size 160px

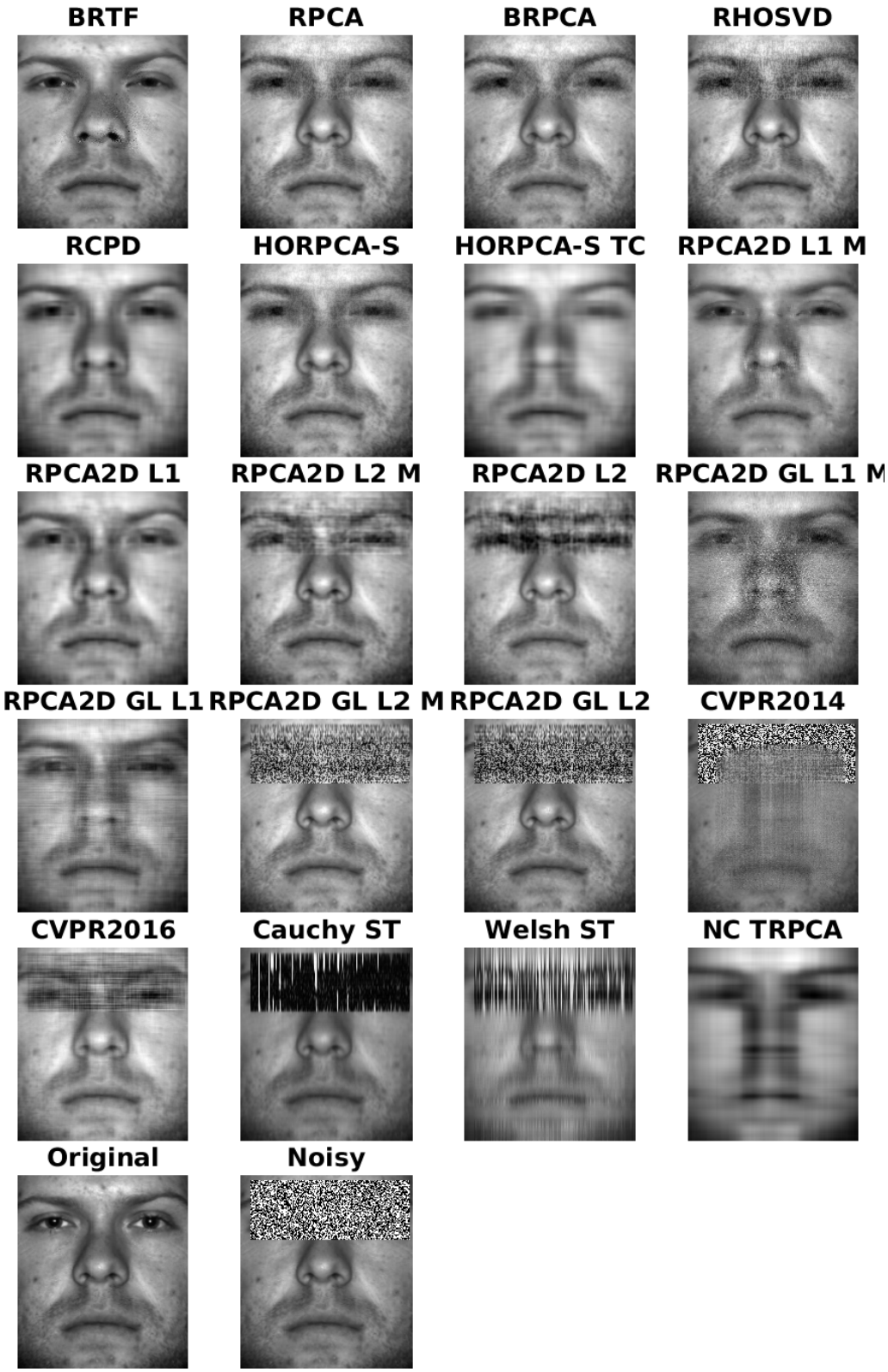

Figure 5.20: Best results per PSNR on Yale-B with patch of max size 160px

Algorithm	Best PSNR	Best SSIM	Best FSIM	Best MSAM
Tensor BRPCA (Papamakarios et. al.)	33.9074	0.954	0.9718	0.0531
Tensor RPCA (Papamakarios et. al.)	32.7486	0.9497	0.9661	0.0635
HORPCA-S	32.5031	0.9448	0.9648	0.0660
Tensor RHOSVD (Papamakarios et. al.)	32.0228	0.9447	0.9632	0.0696
RPCA2D Fro ℓ_2 w/ mean (Bahri et. al.)	29.8869	0.8958	0.9589	0.0987
RPCA2D Fro ℓ_2 (Bahri et. al.)	28.901	0.901	0.9546	0.1141
Tensor RPCA (CVPR 2016)	27.5827	0.836	0.9485	0.1173
HORPCA-S TC	26.6692	0.8833	0.9494	0.1249
RPCA2D Fro ℓ_1 (Bahri et. al.)	26.1636	0.7701	0.9062	0.1444
Cauchy ST	24.6149	0.8798	0.9532	0.1827
Tensor RCPD (Papamakarios et. al.)	24.447	0.7031	0.8706	0.1680
Welsh ST	22.6341	0.8798	0.9435	0.1993
RPCA2D Fro ℓ_1 w/ mean (Bahri et. al.)	22.5729	0.6907	0.8875	0.2114
RPCA2D GL ℓ_2 w/ mean (Bahri et. al.)	21.9948	0.7748	0.9032	0.2161
RPCA2D GL ℓ_2 (Bahri et. al.)	21.829	0.7678	0.9026	0.2209
NC TRPCA	18.6055	0.5546	0.8189	0.3189
Tensor RPCA (CVPR 2014)	18.3076	0.8798	0.8964	0.3151
BRTF	16.8345	0.6117	0.8256	0.3693
RPCA2D GL ℓ_1 (Bahri et. al.)	16.1864	0.6233	0.8512	0.4061
RPCA2D GL ℓ_1 w/ mean (Bahri et. al.)	15.727	0.5668	0.8172	0.4421

Table 5.11: Best performance on Yale-B sorted by PSNR - Patch of max size 100px

Quantitative metrics such as PSNR and FSIM show their limitations on this benchmark, so visual inspection on the results is our preferred way of comparing the algorithms. This issue is made more prominent by the fact that the measures reported in the tables are averaged over the 64 frontal slices of the observation tensor. PSNR and FSIM therefore help identify the best methods but may not always provide an accurate ranking.

Overall, the matricisation-based algorithms perform the best on this patch corruption benchmark, with the exception of the models that use *redescending M-estimators* as their loss functions. *RPCA2D Fro* ℓ_2 is the most competitive 2D model followed by *RPCA2D Fro* ℓ_1, and on this set of results, the robust mean estimator appears to enhance the performance for the ℓ_2 version but to decrease it in certain cases for the ℓ_1 variant. As we already observed in the salt & pepper noise case, *BRTF* tends to strongly homogenise the learnt low-rank components while *NC TRPCA* removes the corruption at the cost of a general degradation of the image quality.

Algorithm	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
Tensor BRPCA (Papamakarios et. al.)	29.4723	0.9044	0.9445	0.0952
Tensor RPCA (Papamakarios et. al.)	28.2569	0.8955	0.936	0.1083
HORPCA-S	28.0179	0.8888	0.9353	0.1087
Tensor RHOSVD (Papamakarios et. al.)	27.6087	0.8869	0.9305	0.1185
RPCA2D Fro ℓ_2 w/ mean (Bahri et. al.)	25.413	0.8083	0.9222	0.1739
Tensor RPCA (CVPR 2016)	25.1943	0.7579	0.9086	0.1605
RPCA2D Fro ℓ_2 (Bahri et. al.)	25.0473	0.8179	0.9188	0.1880
Tensor RCPD (Papamakarios et. al.)	23.4798	0.6803	0.8584	0.1966
HORPCA-S TC	22.1675	0.7613	0.8894	0.2062
RPCA2D Fro ℓ_1 (Bahri et. al.)	21.2769	0.6587	0.8727	0.2409
Cauchy ST	21.1565	0.7716	0.9124	0.2564
Welsh ST	19.7354	0.762	0.8993	0.2853
RPCA2D Fro ℓ_1 w/ mean (Bahri et. al.)	18.698	0.6024	0.8551	0.2901
RPCA2D GL ℓ_2 w/ mean (Bahri et. al.)	18.5108	0.6502	0.8287	0.3075
RPCA2D GL ℓ_2 (Bahri et. al.)	18.4324	0.6767	0.8213	0.3261
BRTF	17.8628	0.5805	0.82	0.3655
NC TRPCA	17.8117	0.4909	0.7799	0.3405
RPCA2D GL ℓ_1 (Bahri et. al.)	16.3034	0.5462	0.7735	0.4393
Tensor RPCA (CVPR 2014)	16.1595	0.762	0.7996	0.3892
RPCA2D GL ℓ_1 w/ mean (Bahri et. al.)	15.4498	0.5226	0.7669	0.4830

Table 5.12: Best performance on Yale-B sorted by PSNR - Patch of max size 160px

Figure 5.21: Original image for the Facade benchmark

5.4 Colour image denoising

Colour images are ubiquitous and generally consist in three colour bands representing the same subject. Therefore, they are well represented by 3-way tensors, and correlation between the bands should be a source of useful information for image denoising.

We test all methods by varying the parameters described in the chapter's introduction and we report four image quality measures. Here, we use FSIMc instead of FSIM as it has been designed specifically for colour images. The SSIM and PSNR reported are the mean values of the SSIMs and PSNRs of each three colour band compared individually and seen as monochromatic images.

Similarly to our tests on the Yale-B dataset, we will test three increasing levels of corruption: 10%, 30%, and 60%. The dataset is the *Facade* image, a photograph of a building's front wall. Figure 5.21 presents the original picture. This image is well-suited for testing low rank recovery because of its highly symmetric and geometric nature. However, the image is also rich in fine details, such as the forged iron ornaments, the light through the glass, or the pattern of the brick wall. Therefore, we believe it is a good benchmark for testing how well fine details can be recovered from noisy low-rank images.

We introduced salt & pepper noise in all three bands and report the performance for each noise level in the form of plots and tables.

At the 10% noise level, most methods provide excellent reconstruction according to both Table 5.13 and Figure 5.22. At this noise level, *HORPCA-S TC*, which treats black noise pixels as missing

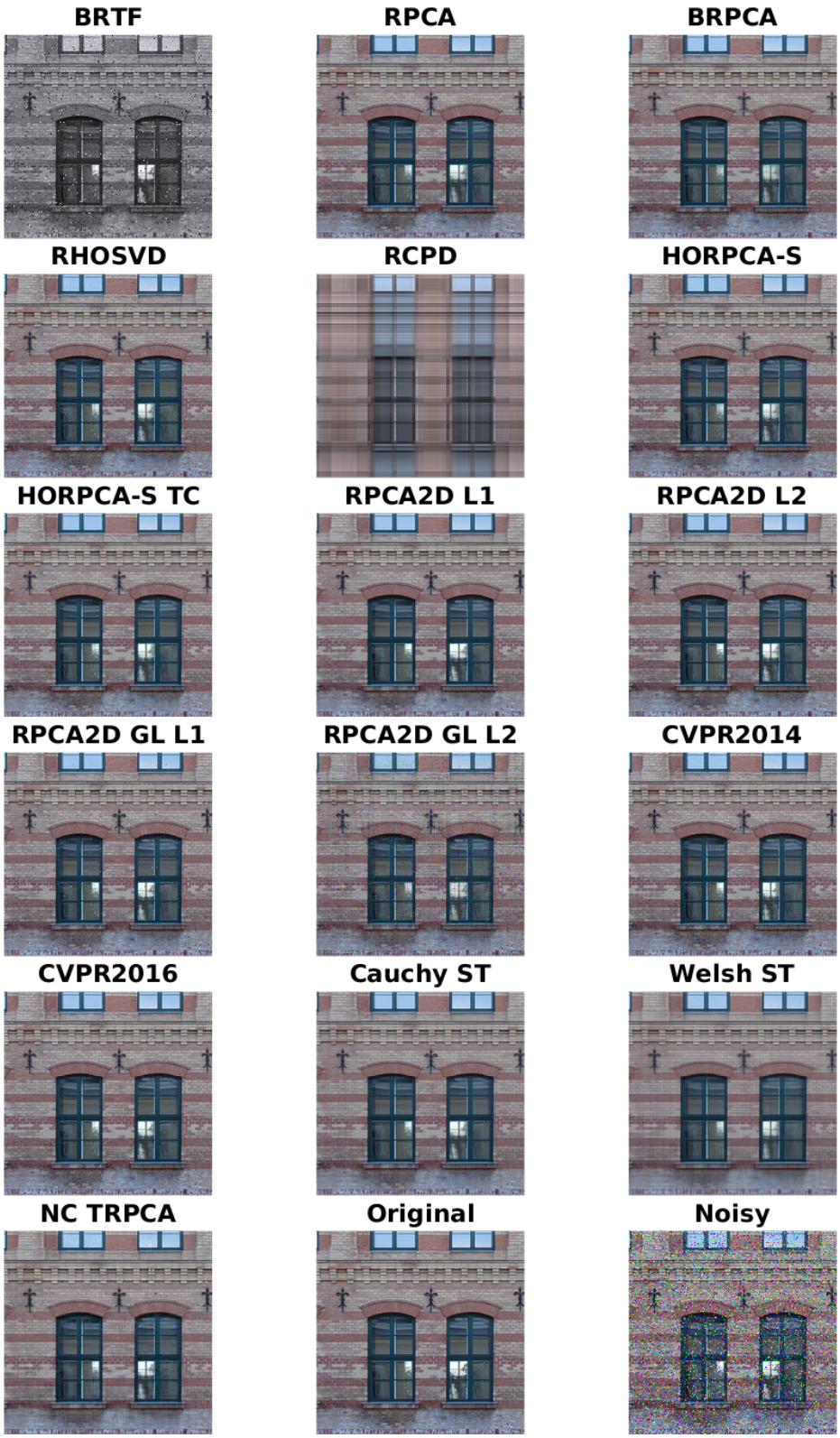

Figure 5.22: Best results per FSIMc on Facade with 10% S & P noise

Algorithm	Best PSNR	Best SSIM	Best FSIMc	Best MSAM
Tensor RPCA (Papamakarios et. al.)	33.1151	0.9608	0.9942	0.0143
Tensor BRPCA (Papamakarios et. al.)	33.0637	0.9650	0.9941	0.0147
Tensor RHOSVD (Papamakarios et. al.)	31.6708	0.9633	0.9926	0.0141
Tensor RCPD (Papamakarios et. al.)	21.2221	0.6001	0.8196	0.0571
RPCA2D Fro ℓ_1 (Bahri et. al.)	33.2469	0.9661	0.9935	0.0214
RPCA2D Fro ℓ_2 (Bahri et. al.)	33.2794	0.9665	0.9959	0.0150
Tensor RPCA (CVPR 2016)	32.7462	0.9735	0.9955	0.0132
BRTF	21.1698	0.7139	0.9326	0.1100
HORPCA-S	33.1370	0.9652	0.9938	0.0144
HORPCA-S TC	33.6646	0.9768	0.9959	0.0115
Tensor RPCA (CVPR 2014)	31.8137	0.9489	0.9886	0.0130
RPCA2D GL ℓ_1 (Bahri et. al.)	29.9740	0.9412	0.9852	0.0260
RPCA2D GL ℓ_2 (Bahri et. al.)	26.3137	0.8231	0.9620	0.0444
Welsh ST	28.0183	0.8971	0.9725	0.0268
Cauchy ST	31.7107	0.9616	0.9924	0.0183
NC TRPCA	27.8527	0.8897	0.9744	0.0334

Table 5.13: Best performance on Facade - 10% Salt & Pepper noise

values, performs slightly better than the standard *HORPCA-S*. The 2D method provided the highest PSNR, the two second-highest SSIMs behind *Tensor RPCA*, and the highest FSIM. The two t-SVD methods [115, 66] provided the lowest MSAMs followed by the methods from Papamakarios [76] and *HORPCA-S*.

Some optimisation-based methods and *BRTF* performed surprisingly low this noise level, such as *Tensor RCPD* and *RPCA2D GL* ℓ_2. Others performed better than on the Yale-B benchmark: *Tensor RPCA (CVPR2014)* provides comparable performance to the best methods, and *NC TRPCA* yields smoother recovered components than on the previous benchmarks.

We didn't sort the tables according to any criterion since we observed a similar phenomenon to the one described in section 5.3.2. The metrics help us determine the set of the best-performing methods but cannot discriminate between them correctly, as will become more apparent at higher noise levels.

At the 30% corruption level, the best methods are the same and *HORPCA-S TC* still performs better than *HORPCA-S*. Surprisingly, *Cauchy ST* performs better than in the 10% noise case according to the quantitative metrics. *Tensor RPCA (CVPR2016)* still provides the second highest FSIMc and among the lowest MSAM, and so does *RPCA 2D Fro* ℓ_2. All best performing methods have close FSIMc and MSAM, but relatively marked differences are observed on PSNR and SSIM.

Visual inspection is again the key to identifying the best algorithms. *HORPCA-S TC* attained a higher PSNR and FSIMc, and lower MSAM scores than any other method, but visual comparison of the images clearly shows many details of the original image are lost: the brick wall is smoother: the ornaments are blurred and the light that comes through the window is dim. Similar comments can be made about *Tensor RPCA (CVPR2016)* (second highest FSIMc), and *Cauchy ST* (second higher

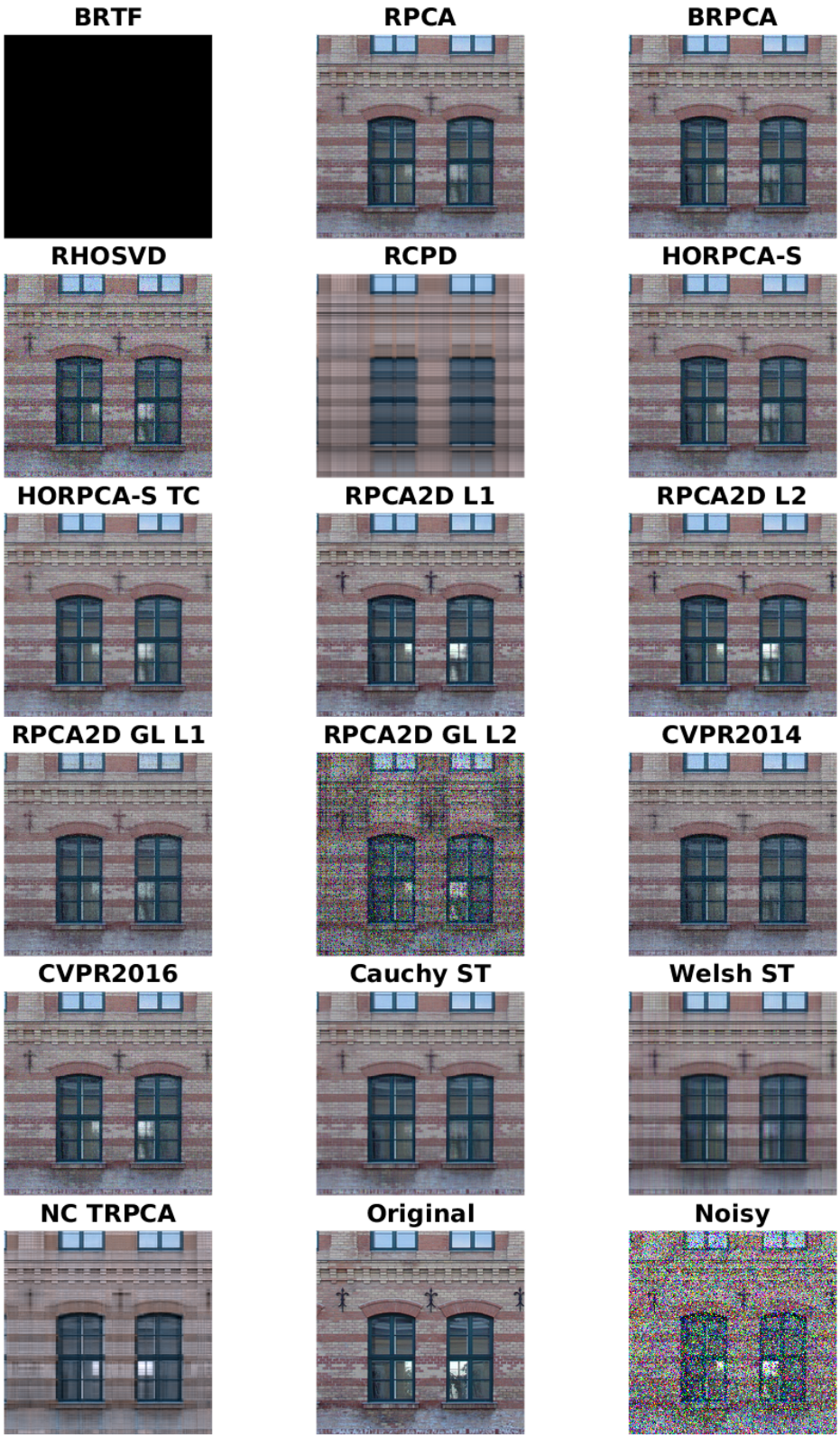

Figure 5.23: Best results per FSIMc on Facade with 30% S & P noise

Algorithm	Best PSNR	Best SSIM	Best FSIMc	Best MSAM
Tensor RPCA (Papamakarios et. al.)	27.7721	0.8791	0.9728	0.0281
Tensor BRPCA (Papamakarios et. al.)	27.4471	0.8674	0.9719	0.0288
Tensor RHOSVD (Papamakarios et. al.)	25.6298	0.8676	0.9536	0.0287
Tensor RCPD (Papamakarios et. al.)	21.2437	0.5982	0.8141	0.0560
RPCA2D Fro ℓ_1 (Bahri et. al.)	27.8261	0.8780	0.9699	0.0344
RPCA2D Fro ℓ_2 (Bahri et. al.)	27.9915	0.8728	0.9741	0.0291
Tensor RPCA (CVPR 2016)	27.0979	0.9004	0.9764	0.0294
BRTF	15.0090	0.2690	0.8036	0.1110
HORPCA-S	27.5200	0.8819	0.9714	0.0282
HORPCA-S TC	29.0687	0.9104	0.9771	0.0256
Tensor RPCA (CVPR 2014)	25.9989	0.8217	0.9484	0.0262
RPCA2D GL ℓ_1 (Bahri et. al.)	26.0848	0.8510	0.9542	0.0403
RPCA2D GL ℓ_2 (Bahri et. al.)	16.6861	0.3218	0.8115	0.0993
Welsh ST	25.2946	0.7886	0.9309	0.0431
Cauchy ST	29.0492	0.9172	0.9724	0.0283
NC TRPCA	21.2992	0.7475	0.8881	0.0646

Table 5.14: Best performance on Facade - 30% Salt & Pepper noise

PSNR). On the contrary, two 2D methods stand out as providing the best reconstruction of all methods: *RPCA2D Fro* ℓ_1 and *RPCA2D Fro* ℓ_2 both produce an image with well preserved colours and the sharpest details. It is striking that the dynamic of the light is much clearly recovered by these two methods than by the others and that the iron ornaments do fade away far less. *NC TRPCA* also preserves a bright light but not the fine details of the picture.

RPCA2D GL ℓ_1's performance quickly degrades as the noise level increases but remains comparable to that of other algorithms such as *Tensor RPCA*, or *Tensor BRPCA*. As with other benchmarks the performance of *RPCA2D GL* ℓ_2 is the lowest of all four 2D methods we propose. *BRTF* failed on this benchmark and we believe finer tuning of its parameters would allow for a better comparison.

In the gross-corruption case, *RPCA2D Fro* ℓ_1 is now the best performer on PSNR and FSIMc, the second best on SSIM behind *Tensor RPCA (CVPR2016)*, and one of the best according to MSAM. The lowest MSAM is attained by *RPCA2D Fro* ℓ_2 which is also second best on PSNR and third best on FSIMc. *Tensor RPCA (CVPR2016)*, *RPCA2D Fro* ℓ_1 and *RPCA2D Fro* ℓ_2's performance stay high while the gap increases with the other well-performing methods. *Cauchy ST*'s quantitative performance remains relatively high but lower than the top methods'. *Tensor RPCA*'s performance on MSAM is second-best, consistent with the lower noise levels. Interestingly, *HORPCA-S TC* performs worse than *HORPCA-S* at this noise level. Visually, *RPCA2D Fro* ℓ_1 demonstrates the highest performance: it is the only method with *Tensor RPCA (CVPR2016)* that preserved both the black iron ornaments and the window light, and did it best in both cases.

Overall, convex optimisation methods dominate this benchmark. *RPCA2D Fro* ℓ_1 and *RPCA2D Fro* ℓ_2 match the performance of the other algorithms at the 10% noise level and outperform all other methods at higher levels with a marked advantage for *RPCA2D Fro* ℓ_1. This suggests that

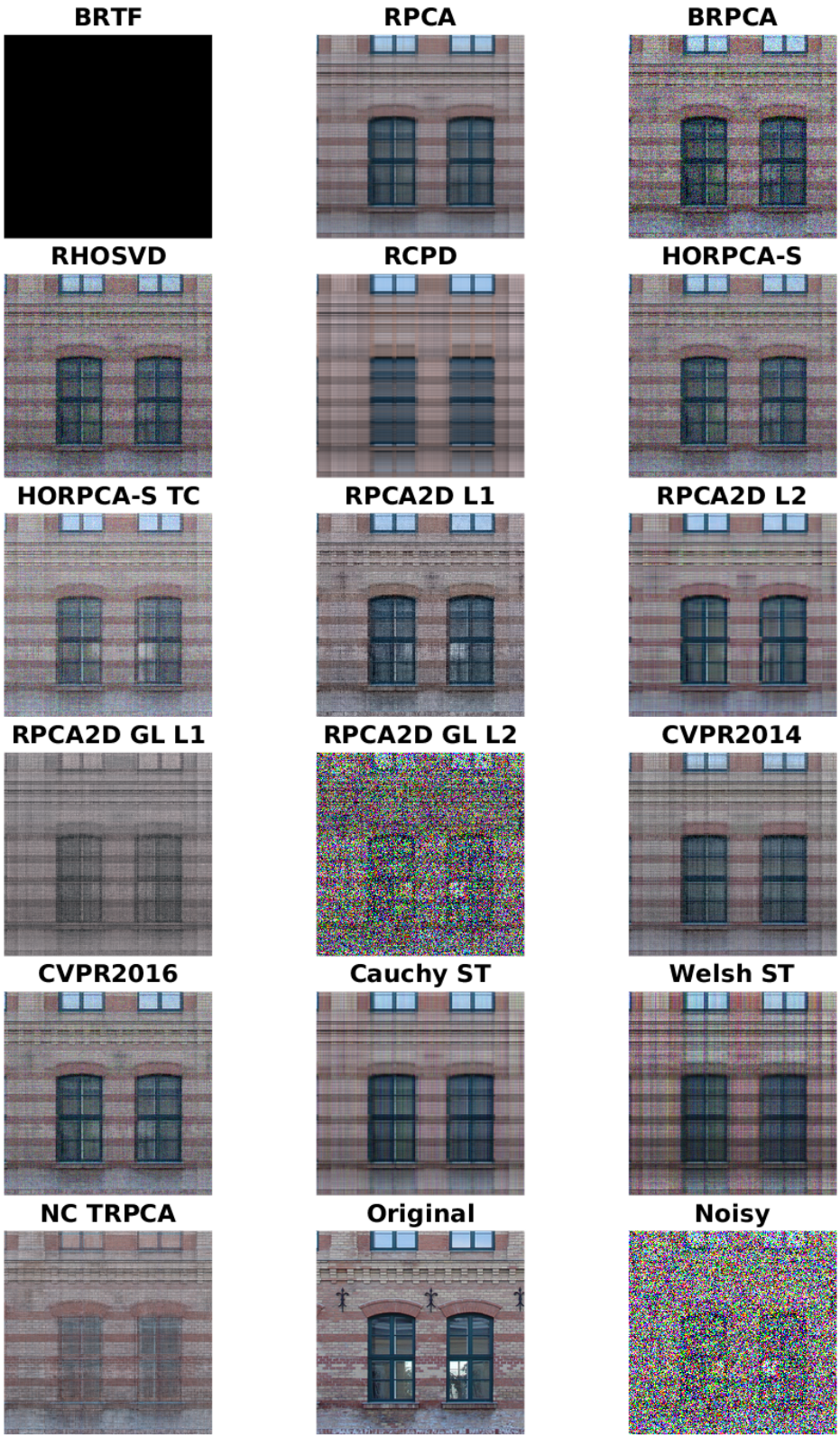

Figure 5.24: Best results per FSIMc on Facade with 60% S & P noise

Algorithm	**Best PSNR**	**Best SSIM**	**Best FSIMc**	**Best MSAM**
Tensor RPCA (Papamakarios et. al.)	22.3366	0.7069	0.8834	0.0447
Tensor BRPCA (Papamakarios et. al.)	22.6003	0.6731	0.8835	0.0645
Tensor RHOSVD (Papamakarios et. al.)	20.5475	0.6455	0.8849	0.0554
Tensor RCPD (Papamakarios et. al.)	21.1616	0.5871	0.8184	0.0598
RPCA2D Fro ℓ_1 (Bahri et. al.)	23.8064	0.7240	0.9152	0.0489
RPCA2D Fro ℓ_2 (Bahri et. al.)	23.7436	0.7050	0.9044	0.0422
Tensor RPCA (CVPR 2016)	23.6552	0.7304	0.9109	0.0485
BRTF	11.9641	0.1118	0.6938	0.1105
HORPCA-S	22.8811	0.7178	0.9060	0.0452
HORPCA-S TC	21.7773	0.6575	0.8717	0.0637
Tensor RPCA (CVPR 2014)	21.4069	0.5851	0.8529	0.0557
RPCA2D GL ℓ_1 (Bahri et. al.)	18.9731	0.5201	0.8156	0.0865
RPCA2D GL ℓ_2 (Bahri et. al.)	14.2515	0.1098	0.6669	0.2017
Welsh ST	19.8508	0.5374	0.8421	0.0903
Cauchy ST	22.3150	0.6840	0.8820	0.0546
NC TRPCA	19.4207	0.6258	0.8121	0.0810

Table 5.15: Best performance on Facade - 60% Salt & Pepper noise

sparsity in the dictionary coefficients improve robustness to noise.

Chapter 6

Basis for a Bayesian treatment of Robust 2D PCA

Inspired by the success of probabilistic approaches in machine learning, we propose a Bayesian model for two-dimensional robust probabilistic PCA. The main advantage we see over the deterministic models is the ability to treat both "hidden variables" and "model parameters"[1] in the same way, *i.e,* inferring all latent variables from the data. In order to make learning and inference tractable, we derive the updates for a Variational Bayesian approximation and discuss how some key quantities can be computed numerically.

6.1 Choice of priors

In this section we refer to the matrix observations as $\mathbf{Y}_n$. The set of all the observations for a 3-way tensor $\boldsymbol{\mathcal{Y}}$ where each frontal slice is one observation.

It has been shown that the ℓ_2 is equivalent to a Gaussian prior while the sparsity-inducing ℓ_1 is equivalent to a Laplace prior on the relevant variables [73]. In the following subsections, we describe the model assumptions on the priors of the variables.

The observation model is:

$$\forall n, \quad \mathbf{Y}_n = \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} + \mathbf{E}_n + \boldsymbol{\epsilon} \tag{6.1}$$

Where $\mathbf{E}_n$ represents a sparse-error term of arbitrarily large magnitude (*i.e,* Laplacian noise), and $\boldsymbol{\epsilon}$ a dense but low-magnitude noise.

We can also write a more general model in tensor form:

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{T}} \times_1 \mathbf{U_c} \times_2 \mathbf{U_r} + \boldsymbol{\mathcal{E}} + \epsilon \tag{6.2}$$

Equation 6.1 corresponds to a special case of 6.2 where we assume that all the matrix-valued observations for the frontal slices of the tensor and are *independent and identically distributed*.

6.1.1 Sparse error

We do not assume any structure in the sparse noise of the observations, so the noise elements are all independent and the following factorisation holds:

$$p(\mathbf{E}_n) = \prod_{i,j} p(e^n_{i,j})$$

[1] Quotations on purpose

And more generally:

$$p(\boldsymbol{\mathcal{E}}) = \prod_{n,i,j} p(e_{i,j}^n)$$

For tractability of the model, we replace the Laplace prior that would be strictly equivalent to the ℓ_1 by a two-stage hierarchical model:

$$p(e_{i,j}^n) = \mathcal{N}(e_{i,j}^n|0, {\alpha_{i,j}^n}^{-1})p({\alpha_{i,j}^n}^{-1})$$

The prior on the precision of the 0-mean Gaussian can be chosen to induce sparsity. Guan and Dy [44] have studied three possible priors: exponential, inverse Gaussian, and Jeffreys' prior and have shown that Jeffreys' prior yields the sparsest solutions. Jeffreys' prior is the one used in Babacan et al. [6] and is the prior we chose to use as well:

$$p({\alpha_{i,j}^n}^{-1}) \propto \frac{1}{{\alpha_{i,j}^n}^{-1}}$$

This choice means that the distributions of each ${\alpha_{i,j}^n}^{-1}$ is log-uniform, therefore we expect that ${\alpha_{i,j}^n}^{-1}$'s order of magnitude matters more than its exact value.

6.1.2 Latent core tensor

In our model, $\mathbf{T}_n \in \mathbb{R}^{r\times r}$. We assume a matrix-normal distribution with identity row and column covariance on $\mathbf{T}_n$ since there is a non-identifiability between $\mathbf{U_c}$ and the column-covariance and between $\mathbf{U_r}$ and the row-covariance. We chose not to assume a mean of $\mathbf{0}$ since there is no separate global observation mean for the moment, we leave this extension to future work.

$$p(\mathbf{T}_n) = \mathcal{N}(\mathbf{T}_n|\mathbf{M}, \mathbf{I}_r, \mathbf{I}_r) = (2\pi)^{\frac{-r^2}{2}} \exp\left[\mathrm{tr}\left[-\frac{1}{2}(\mathbf{T}_n - \mathbf{M})(\mathbf{T}_n - \mathbf{M})^\mathsf{T}\right]\right]$$

6.1.3 Left and right bases

We assume $\mathbf{U_c} \in \mathbb{R}^{n\times r}$ and $\mathbf{U_r} \in \mathbb{R}^{m\times r}$. We also assume column-independence within $\mathbf{U_c}$ and $\mathbf{U_r}$ and that each column $\mathbf{u}_{ci}$ and $\mathbf{u}_{ri}$ follows a multivariate Gaussian distribution with $\mathbf{0}$ mean and precision $\boldsymbol{\gamma}_i$:

$$p(\mathbf{U_c}) = \prod_i p(\mathbf{u}_{ci}) = \prod_i \mathcal{N}(\mathbf{u}_{ci}|\mathbf{0}, \boldsymbol{\gamma}_i^{-1}\mathbf{I}_n)$$
$$p(\mathbf{U_r}) = \prod_i p(\mathbf{u}_{ri}) = \prod_i \mathcal{N}(\mathbf{u}_{ri}|\mathbf{0}, \boldsymbol{\gamma}_i^{-1}\mathbf{I}_m)$$

The common precisions $\boldsymbol{\gamma}_i$ models the correlation between the columns of $\mathbf{U_c}$ and $\mathbf{U_r}$.

6.1.4 Distribution of each $\boldsymbol{\gamma}_i$

We assume a Gamma distribution of parameters a and b on each $\boldsymbol{\gamma}_i$:

$$\forall i, \quad p(\boldsymbol{\gamma}_i) = \mathrm{Ga}(a, b) = \frac{1}{\boldsymbol{\gamma}(a)b^a}\boldsymbol{\gamma}_i^{a-1}e^{-\frac{\boldsymbol{\gamma}_i}{b}}$$

6.1.5 White noise and distribution of $\mathbf{Y}_n$

ϵ is a white noise term that gives $\mathbf{Y}_n$ a normal distribution. It follows a matrix-normal distribution of mean 0 and covariances directed by a precision parameter β:

$$p(\mathbf{Y}_n) \propto \exp\left[-\frac{\beta}{2}||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2\right]$$

We give β Jeffreys' prior:

$$p(\beta) \propto \frac{1}{\beta}$$

One possible choice for the explicit matrix-normal is $\mathcal{N}(\mathbf{Y}_n|\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} + \mathbf{E}_n, \beta^{-\frac{1}{2}}\mathbf{I}_n, \beta^{-\frac{1}{2}}\mathbf{I})$, but this is not unique nor required.

6.2 Joint distribution and graphical model

The joint distribution is given by:

$$p(\mathbf{Y}, \mathbf{T}, \mathbf{U_c}, \mathbf{U_r}, \mathbf{E}, \boldsymbol{\alpha}, \beta, \boldsymbol{\gamma})$$

And factorises as:

$$p(\mathbf{Y}|\mathbf{T}, \mathbf{U_c}, \mathbf{U_r}, \mathbf{E}, \boldsymbol{\alpha}, \beta, \boldsymbol{\gamma})p(\mathbf{E}|\boldsymbol{\alpha})p(\mathbf{U_c}|\boldsymbol{\gamma})p(\mathbf{U_r}|\boldsymbol{\gamma})p(\mathbf{T})p(\boldsymbol{\alpha})p(\beta)p(\boldsymbol{\gamma})$$

We further assume that the data and the latent variables are independent and identically distributed, this gives the final factorisation which we also represent as a graphical model in Figure 6.1:

$$\prod_n [\, p(\mathbf{Y}_n|\mathbf{T}_n, \mathbf{E}_n, \mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\alpha}, \beta, \boldsymbol{\gamma})p(\mathbf{E}_n|\boldsymbol{\alpha})p(\mathbf{T}_n)\,]\, p(\mathbf{U_c}|\boldsymbol{\gamma})p(\mathbf{U_r}|\boldsymbol{\gamma})p(\boldsymbol{\alpha})p(\beta)p(\boldsymbol{\gamma})$$

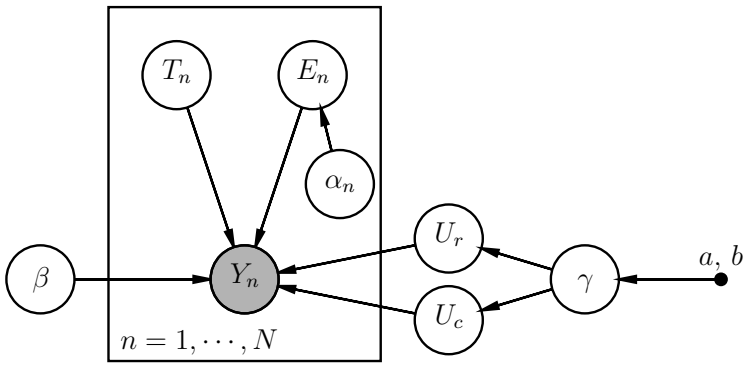

Figure 6.1: Graphical model of our Bilinear Probabilistic Robust PCA

6.3 Solution with variational inference

6.3.1 Logarithms of the distributions

We denote by $\mathbf{Z}$ the set of latent variables (including the so-called model parameters). Recall the variational solution that maximises the evidence lower-bound is

$$\log q^*(z_i) = \langle \log p(\mathbf{Y}, \mathbf{Z}) \rangle_{\mathbf{Z} \setminus z_i} \tag{6.3}$$

We list the logarithms of the important distributions thereafter:

$$\log p(\alpha_{i,j}^n) = -\log p(\alpha_{i,j}^n) \tag{6.4}$$

$$\log p(\beta) = -\log p(\beta) \tag{6.5}$$

$$\log p(\boldsymbol{\gamma}_i|a,b) = -\log(\boldsymbol{\gamma}(a)) - a\log b + (a-1)\log\boldsymbol{\gamma}_i - \frac{\boldsymbol{\gamma}_i}{b} \tag{6.6}$$

$$\log p(\mathbf{u}_{ci}|\boldsymbol{\gamma}_i) = -\frac{n}{2}\log 2\pi + \frac{n}{2}\log\boldsymbol{\gamma}_i - \frac{1}{2}\boldsymbol{\gamma}_i\mathbf{u}_{ci}^{\mathsf{T}}\mathbf{u}_{ci} \tag{6.7}$$

$$\log p(\mathbf{u}_{ri}|\boldsymbol{\gamma}_i) = -\frac{m}{2}\log 2\pi + \frac{m}{2}\log\boldsymbol{\gamma}_i - \frac{1}{2}\boldsymbol{\gamma}_i\mathbf{u}_{ci}^{\mathsf{T}}\mathbf{u}_{ri} \tag{6.8}$$

$$\log p(e_{i,j}^n|\alpha_{i,j}^n) = \log\alpha_{i,j}^n - \frac{1}{2}\log 2\pi - \frac{1}{2}\alpha_{i,j}^n\left(e_{i,j}^n\right)^2 \tag{6.9}$$

$$\log p(\mathbf{T}_n) = -\frac{r^2}{2}\log 2\pi - \frac{1}{2}\mathrm{tr}\left[(\mathbf{T}_n - \mathbf{M})(\mathbf{T}_n - \mathbf{M})^{\mathsf{T}}\right] \tag{6.10}$$

$$\log p(\mathbf{Y}_n|\ldots) = -\frac{\beta}{2}||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}} - \mathbf{E}_n||_{\mathrm{F}}^2 - \frac{1}{2}mn\log 2\pi + \frac{1}{2}mn\log\beta \tag{6.11}$$

We assume a fully factorised variational approximation (unstructured mean field): $q(\mathbf{Z}) = \prod_i q(\mathbf{z_i})$. We also assume independence between the different $\boldsymbol{\gamma}_i$:

$$q(\boldsymbol{\gamma}) = \prod_i q(\boldsymbol{\gamma}_i)$$

And taking in account the unstructured noise and iid data and latent variables assumptions, we find that the logarithm of the joint distribution is given by:

$$\begin{aligned}\log p(\mathbf{Y},\mathbf{Z}) = &\sum_n \log p(\mathbf{Y}_n|\ldots) + \sum_n\sum_i\sum_j \log p(e_{i,j}^n|\alpha_{i,j}^n) + \sum_n \log p(\mathbf{T}_n) + \sum_i \log p(\mathbf{u}_{ci}|\boldsymbol{\gamma}_i) \\ &+ \sum_i \log p(\mathbf{u}_{ri}|\boldsymbol{\gamma}_i) + \sum_i p(\boldsymbol{\gamma}_i) + \sum_i\sum_j p(\alpha_{i,j}^n) + \log\beta\end{aligned} \tag{6.12}$$

6.3.2 Variational updates

Update equation for $e_{i,j}^n$

$$\begin{aligned}\log q*(e_{i,j}^n) &= \langle -\frac{1}{2}\alpha_{i,j}^n\left(e_{i,j}^n\right)^2\rangle - \left[\langle\frac{\beta}{2}||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}} - \mathbf{E}_n||_{\mathrm{F}}^2\rangle\right]_{i,j} + \text{const} \\ &= \langle -\frac{1}{2}\alpha_{i,j}^n\left(e_{i,j}^n\right)^2\rangle - \langle\frac{\beta}{2}(\mathbf{Y}_{i,j}^n - (\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}})_{i,j} - e_{i,j}^n)^2\rangle + \text{const} \\ &= -\frac{1}{2}(\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle)\left(e_{i,j}^n\right)^2 + e_{i,j}^n\langle\beta(\mathbf{Y}_{i,j}^n - (\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}})_{i,j})\rangle + \text{const} \\ &= -\frac{1}{2}(\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle)\left(e_{i,j}^n\right)^2 + e_{i,j}^n\langle\beta\rangle(\mathbf{Y}_{i,j}^n - \langle(\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}})_{i,j}\rangle) + \text{const} \\ &= \frac{1}{2}\left(e_{i,j}^n - \frac{\langle\beta\rangle}{\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle}(\mathbf{Y}_{i,j}^n - \langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}}\rangle_{i,j})\right)^2(\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle) \\ &\quad - \frac{1}{2}(\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle)\langle\beta\rangle^2(\mathbf{Y}_{i,j}^n - \langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^{\mathsf{T}}\rangle_{i,j})^2 + \text{const}\end{aligned}$$

$$= \frac{1}{2}\left(e_{i,j}^n - \frac{\langle\beta\rangle}{\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle}(\mathbf{Y}_{i,j}^n - (\langle\mathbf{U_c}\rangle\langle\mathbf{T}_n\rangle\langle\mathbf{U_r}\rangle^\mathsf{T})_{i,j})\right)^2 (\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle) + \text{const}$$

Using the independence assumptions and in particular $\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\rangle = \langle\mathbf{U_c}\rangle\langle\mathbf{T}_n\rangle\langle\mathbf{U_r}\rangle^\mathsf{T}$. Therefore:

$$q^*(e_{i,j}^n) = \mathcal{N}\left(\frac{\langle\beta\rangle}{\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle}(\mathbf{Y}_{i,j}^n - (\langle\mathbf{U_c}\rangle\langle\mathbf{T}_n\rangle\langle\mathbf{U_r}\rangle^\mathsf{T})_{i,j}), \frac{1}{\langle\alpha_{i,j}^n\rangle + \langle\beta\rangle}\right) \tag{6.13}$$

Update equation for $\mathbf{T}_n$

$$\begin{aligned}\log q^*(\mathbf{T}_n) &= \langle -\frac{1}{2}\text{tr}\left[(\mathbf{T}_n - \mathbf{M})(\mathbf{T}_n - \mathbf{M})^\mathsf{T}\right] - \frac{\beta}{2}||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\text{F}^2\rangle + \text{const} \\ &= -\frac{1}{2}\text{tr}\left[(\mathbf{T}_n - \mathbf{M})(\mathbf{T}_n - \mathbf{M})^\mathsf{T}\right] - \frac{\langle\beta\rangle}{2}\langle||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\text{F}^2\rangle + \text{const}\end{aligned}$$

It is interesting to note that:

$$\langle\beta\rangle\langle||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\text{F}^2\rangle = \frac{\langle||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\text{F}^2\rangle}{\sum_k\langle||\mathbf{Y}_k - \mathbf{U_c}\mathbf{T}_k\mathbf{U_r}^\mathsf{T} - \mathbf{E}_k||_\text{F}^2\rangle} \tag{6.14}$$

Expanding $\langle||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\text{F}^2\rangle$:

$$\begin{aligned}&||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n||_\text{F}^2 = \text{tr}\left[\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n)(\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T} - \mathbf{E}_n)^\mathsf{T}\right] \\ &= \text{tr}[\mathbf{Y}_n\mathbf{Y}_n^\mathsf{T} - \mathbf{Y}_n\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T} - \mathbf{Y}_n\mathbf{E}_n^\mathsf{T} - \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{Y}_n^\mathsf{T} + \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T} + \mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{E}_n^\mathsf{T} \\ &- \mathbf{E}_n\mathbf{Y}_n^\mathsf{T} + \mathbf{E}_n\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T} + \mathbf{E}_n\mathbf{E}_n^\mathsf{T}]\end{aligned}$$

We now use the following identity: $\text{tr}\left[\mathbf{A}^\mathsf{T}\right] = \text{tr}\left[\mathbf{A}\right]$ and the linearity of the trace to get, keeping only the terms in $\mathbf{T}_n$:

$$\begin{aligned}\log q^*(\mathbf{T}_n) = -\frac{1}{2}\text{tr}\Big[&\mathbf{T}_n\mathbf{T}_n^\mathsf{T} - 2\mathbf{T}_n\mathbf{M}^\mathsf{T} + \langle\beta\rangle(-2\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\rangle + 2\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{E}_n^\mathsf{T}\rangle + \\ &\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T}\rangle)\Big] + \text{const}\end{aligned}$$

Breaking down the computations:

$$\begin{aligned}\text{tr}\left[\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\rangle\right] &= \langle\text{tr}\left[\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\right]\rangle \\ &= \langle\text{tr}\left[\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\mathbf{U_c}\right]\rangle \\ &= \text{tr}\left[\mathbf{T}_n\langle\mathbf{U_r}^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\mathbf{U_c}\rangle\right] \\ &= \text{tr}\left[\mathbf{T}_n\langle\mathbf{U_r}\rangle^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\langle\mathbf{U_c}\rangle\right]\end{aligned}$$

Here we made use of the invariance of the trace by circular permutation, and the linearity of the trace, expectation, and transposition operators.

Similarly:

$$\text{tr}\left[\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{E}_n^\mathsf{T}\rangle\right] = \text{tr}\left[\mathbf{T}_n\langle\mathbf{U_r}\rangle^\mathsf{T}\langle\mathbf{E}_n\rangle^\mathsf{T}\langle\mathbf{U_c}\rangle\right]$$

Which allows for some intermediary result:

$$\begin{aligned}\log q^*(\mathbf{T}_n) = &-\frac{1}{2}\mathrm{tr}\Big[\mathbf{T}_n\mathbf{T}_n^\mathsf{T} + \langle\beta\rangle\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T}\rangle - \\ &2\mathbf{T}_n\Big(\mathbf{M}^\mathsf{T} + \langle\beta\rangle\langle\mathbf{U_r}\rangle^\mathsf{T}\mathbf{Y}_n^\mathsf{T}\langle\mathbf{U_c}\rangle - \langle\beta\rangle\langle\mathbf{U_r}\rangle^\mathsf{T}\langle\mathbf{E}_n\rangle^\mathsf{T}\langle\mathbf{U_c}\rangle\Big)\Big] \\ &+ \mathrm{const} \\ = &-\frac{1}{2}\mathrm{tr}\Big[\mathbf{T}_n\mathbf{T}_n^\mathsf{T} + \langle\beta\rangle\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T}\rangle - \\ &2\mathbf{T}_n\Big(\mathbf{M} + \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\mathbf{Y}_n\langle\mathbf{U_r}\rangle - \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\langle\mathbf{E}_n\rangle\langle\mathbf{U_r}\rangle\Big)^\mathsf{T}\Big] \\ &+ \mathrm{const}\end{aligned}$$

We further re-arrange the terms:

$$\begin{aligned}\mathrm{tr}\Big[\langle\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T}\rangle\Big] &= \langle\mathrm{tr}\Big[\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\mathbf{U_c}^\mathsf{T}\Big]\rangle \\ &= \langle\mathrm{tr}\Big[\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\Big]\rangle \\ &= \mathrm{tr}\Big[\langle\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\rangle\Big]\end{aligned}$$

Noting that $\langle\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\mathbf{T}_n\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\mathbf{T}_n^\mathsf{T}\rangle = \langle\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\rangle\mathbf{T}_n\langle\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\rangle\mathbf{T}_n^\mathsf{T}$ we get that:

$$\begin{aligned}\log q^*(\mathbf{T}_n) = &-\frac{1}{2}\mathrm{tr}\Big[\mathbf{T}_n\mathbf{T}_n^\mathsf{T} - 2\mathbf{T}_n\Big(\mathbf{M} + \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\mathbf{Y}_n\langle\mathbf{U_r}\rangle - \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\langle\mathbf{E}_n\rangle\langle\mathbf{U_r}\rangle\Big)^\mathsf{T}\Big] \\ &- \frac{1}{2}\mathrm{tr}\Big[\langle\beta\rangle\langle\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\rangle\mathbf{T}_n\langle\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\rangle\mathbf{T}_n^\mathsf{T}\Big] + \mathrm{const}\end{aligned} \tag{6.15}$$

By expanding the general term inside the trace of a matrix-valued normal distribution we get:

$$\mathrm{tr}\Big[\mathbf{C}^{-1}(\mathbf{X}-\mathbf{W})\mathbf{R}^{-1}(\mathbf{X}-\mathbf{W})^\mathsf{T}\Big] = \mathrm{tr}\Big[\mathbf{C}^{-1}\mathbf{X}\mathbf{R}^{-1}\mathbf{X}^\mathsf{T}\Big] - 2\mathrm{tr}\Big[\mathbf{C}^{-1}\mathbf{X}\mathbf{R}^{-1}\mathbf{W}^\mathsf{T}\Big] + \mathrm{tr}\Big[\mathbf{C}^{-1}\mathbf{W}\mathbf{R}^{-1}\mathbf{W}^\mathsf{T}\Big]$$

We recognise that (6.15) is the product of two matrix-valued normal distributions:

$$\begin{aligned}G_1: \quad &\boldsymbol{\mu}_1 = \mathbf{M} + \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\mathbf{Y}_n\langle\mathbf{U_r}\rangle - \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\langle\mathbf{E}_n\rangle\langle\mathbf{U_r}\rangle \\ &\boldsymbol{\Sigma}_{R,1}^{-1} = \boldsymbol{\Sigma}_{C,1}^{-1} = \mathbf{I} \\ G_2: \quad &\boldsymbol{\mu}_2 = \mathbf{0} \\ &\boldsymbol{\Sigma}_{R,2}^{-1} = \sqrt{\langle\beta\rangle}\langle\mathbf{U_r}^\mathsf{T}\mathbf{U_r}\rangle \\ &\boldsymbol{\Sigma}_{C,2}^{-1} = \sqrt{\langle\beta\rangle}\langle\mathbf{U_c}^\mathsf{T}\mathbf{U_c}\rangle\end{aligned}$$

Note that for G_2 any combination of $\boldsymbol{\Sigma}_{R,2}^{-1}$ and $\boldsymbol{\Sigma}_{C,2}^{-1}$ that leads to a $\langle\beta\rangle$ factor seems acceptable.

These two matrix-valued distributions are equivalent to the following multivariate normal distributions:

$$\begin{aligned}G_1' :&\boldsymbol{\mu}_1' = \mathrm{vec}\Big(\mathbf{M} + \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\mathbf{Y}_n\langle\mathbf{U_r}\rangle - \langle\beta\rangle\langle\mathbf{U_c}\rangle^\mathsf{T}\langle\mathbf{E}_n\rangle\langle\mathbf{U_r}\rangle\Big) \\ &\boldsymbol{\Sigma}^{-1} = \mathbf{I} \\ G_2' :&\boldsymbol{\mu}_2' = \mathbf{0}\end{aligned}$$

$$\mathbf{\Sigma}_2^{-1} = \langle\beta\rangle\langle {\mathbf{U_r}}^\mathsf{T}\mathbf{U_r}\rangle \otimes \langle {\mathbf{U_c}}^\mathsf{T}\mathbf{U_c}\rangle$$

And the mean and covariance of the product are given by:

$$\mathbf{\Sigma} = \left(\mathbf{I}_{r^2} + \langle\beta\rangle\langle {\mathbf{U_r}}^\mathsf{T}\mathbf{U_r}\rangle \otimes \langle {\mathbf{U_c}}^\mathsf{T}\mathbf{U_c}\rangle\right)^{-1} \tag{6.16}$$

$$\mu = \mathbf{\Sigma}\boldsymbol{\mu}_1' \tag{6.17}$$

We explored one possible way of computing the inverse of Σ:
$\mathbf{U_c}^\mathsf{T}\mathbf{U_c}$ and $\mathbf{U_r}^\mathsf{T}\mathbf{U_r}$ are symmetric real-valued matrices so it follows from the spectral theorem that there exist two orthogonal bases $\mathbf{P}_C$ $\mathbf{P}_R$ such that:

$$\mathbf{U_c}^\mathsf{T}\mathbf{U_c} = \mathbf{P}_C\mathbf{\Lambda}_C\mathbf{P}_C^\mathsf{T} \quad \text{and} \quad \mathbf{U_r}^\mathsf{T}\mathbf{U_r} = \mathbf{P}_R\mathbf{\Lambda}_R\mathbf{P}_R^\mathsf{T}$$

Using the compatibility of the Kronecker product with the eigenvalue decomposition and its inversion/transposition properties we get that:

$$\left(\mathbf{I}_{r^2} + \langle\beta\rangle\langle {\mathbf{U_r}}^\mathsf{T}\mathbf{U_r}\rangle \otimes \langle {\mathbf{U_c}}^\mathsf{T}\mathbf{U_c}\rangle\right) = (\mathbf{P}_R \otimes \mathbf{P}_C)(\mathbf{I} + \mathbf{\Lambda}_R \otimes \mathbf{\Lambda}_C)(\mathbf{P}_R \otimes \mathbf{P}_C)^\mathsf{T}$$

and the inverse is

$$(\mathbf{P}_R \otimes \mathbf{P}_C)(\mathbf{I} + \mathbf{\Lambda}_R \otimes \mathbf{\Lambda}_C)^{-1}(\mathbf{P}_R \otimes \mathbf{P}_C)^\mathsf{T}$$

However this is not computationally efficient, and the best way we found so far is to put the system formed by equations 6.16 and 6.17 into a Stein equation (see section 4.3.3 equations (4.3) and (4.4)) and solve it via the *Hessenberg-Schur* method.

Update equation for $\mathbf{U_c}$

Let $\mathbf{W}_n = \mathbf{U_r}\mathbf{T}_n^\mathsf{T}$. As usual we take the expectation of the terms where $\mathbf{U_c}$ appears with respect to every other variable (or equivalently to the variables in $\mathbf{U_c}$'s Markov blanket):

$$\begin{aligned}
\log q*(\mathbf{U_c}) &= \langle\sum_i \log p(\mathbf{u}_{ci}|\boldsymbol{\gamma}_i) + \frac{\langle\beta\rangle}{2}\sum_n ||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2\rangle + \text{const} \\
&= \langle\sum_i -\frac{1}{2}\langle\boldsymbol{\gamma}_i\rangle\mathbf{u}_{ci}^\mathsf{T}\mathbf{u}_{ci} + \frac{\langle\beta\rangle}{2}\sum_n ||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2\rangle + \text{const} \\
&= \langle -\frac{1}{2}\mathrm{tr}\left[{\mathbf{U_c}}^\mathsf{T}\boldsymbol{\gamma}\mathbf{U_c}\right] + \frac{\langle\beta\rangle}{2}\sum_n ||\mathbf{Y}_n - \mathbf{U_c}\mathbf{T}_n{\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2\rangle + \text{const} \\
&= \langle -\frac{1}{2}\mathrm{tr}\left[{\mathbf{U_c}}^\mathsf{T}\boldsymbol{\gamma}\mathbf{U_c}\right] + \frac{\langle\beta\rangle}{2}\sum_n ||\mathbf{Y}_n - \mathbf{U_c}\mathbf{W}_n^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2\rangle + \text{const}
\end{aligned}$$

Expanding the squared Frobenius norm and keeping only the terms in $\mathbf{U_c}$:

$$\begin{aligned}
\log q*(\mathbf{U_c}) &= \langle -\frac{1}{2}\mathrm{tr}\left[{\mathbf{U_c}}^\mathsf{T}\boldsymbol{\gamma}\mathbf{U_c}\right] + \frac{\langle\beta\rangle}{2}\sum_n \mathrm{tr}\left[\mathbf{U_c}\mathbf{W}_n^\mathsf{T}\mathbf{W}_n{\mathbf{U_c}}^\mathsf{T} - 2\mathbf{U_c}\mathbf{W}_n^\mathsf{T}\mathbf{Y}_n^\mathsf{T} + 2\mathbf{U_c}\mathbf{W}_n^\mathsf{T}\mathbf{E}_n^\mathsf{T}\right]\rangle \\
&\quad + \text{const} \\
&= \langle -\frac{1}{2}\mathrm{tr}\left[{\mathbf{U_c}}^\mathsf{T}\boldsymbol{\gamma}\mathbf{U_c}\right] + \frac{\langle\beta\rangle}{2}\sum_n \mathrm{tr}\left[\mathbf{U_c}\mathbf{W}_n^\mathsf{T}\mathbf{W}_n{\mathbf{U_c}}^\mathsf{T} - 2\mathbf{U_c}\mathbf{W}_n^\mathsf{T}(\mathbf{Y}_n - \mathbf{E}_n)^\mathsf{T}\right]\rangle + \text{const}
\end{aligned}$$

From which we identify the parameters of a matrix-valued Normal distribution:

$$\Sigma_{\mathbf{U_c}} = \left(\boldsymbol{\gamma} + \langle \beta \rangle \sum_n \langle \mathbf{W}_n^\mathsf{T} \mathbf{W}_n \rangle \right)^{-1}$$
$$\mu_{\mathbf{U_c}}^\mathsf{T} = \langle \beta \rangle \Sigma_{\mathbf{U_c}} \left(\sum_n (\mathbf{Y}_n - \mathbf{E}_n) \mathbf{W}_n \right)^\mathsf{T}$$

Update equation for $\mathbf{U_r}$

With an equivalent calculation by setting $\mathbf{W}_n = \mathbf{U_c} \mathbf{T}_n$ we get a matrix-valued Normal distribution:

$$\Sigma_{\mathbf{U_r}} = \left(\boldsymbol{\gamma} + \langle \beta \rangle \sum_n \langle \mathbf{W}_n^\mathsf{T} \mathbf{W}_n \rangle \right)^{-1}$$
$$\mu_{\mathbf{U_r}}^\mathsf{T} = \langle \beta \rangle \Sigma_{\mathbf{U_r}} \left(\sum_n \mathbf{W}_n^\mathsf{T} (\mathbf{Y}_n - \mathbf{E}_n) \right)$$

Update equation for $\alpha_{i,j}^n$

$$\begin{aligned} \log q^*(\alpha_{i,j}^n) &= \langle \log p(\mathbf{Y} | \dots \alpha_{i,j}^n \dots) + \log(p(\alpha_{i,j}^n) + \log p(\mathbf{E} | \alpha_{i,j}^n) \rangle_{\mathbf{Z} \setminus \alpha_{i,j}^n} + \text{const} \\ &= \langle \frac{1}{2} \alpha_{i,j}^n \left(e_{i,j}^n \right)^2 + \log \alpha_{i,j}^n - \log \alpha_{i,j}^n \rangle + \text{const} \\ &= (1-1) \log \alpha_{i,j}^n - \alpha_{i,j}^n \langle \left(e_{i,j}^n \right)^2 \rangle + \text{const} \end{aligned}$$

Recall $\langle X^2 \rangle = \text{var} X + \langle X \rangle^2$:

$$\log q^*(\alpha_{i,j}^n) = (1-1) \log \alpha_{i,j}^n - \alpha_{i,j}^n \left(\Sigma_{e_{i,j}^n} + \langle e_{i,j}^n \rangle^2 \right) + \text{const}$$

So:

$$q^*(\alpha_{i,j}^n) = \text{Ga} \left(1, \frac{1}{\left(\Sigma_{e_{i,j}^n} + \langle e_{i,j}^n \rangle^2 \right)} \right) \tag{6.18}$$

Update equation for β

$$\begin{aligned} \log q^*(\beta) &= \langle \sum_k \frac{1}{2} mn \log \beta - \frac{\beta}{2} ||\mathbf{Y}_k - \mathbf{U_c} \mathbf{T}_k \mathbf{U_r}^\mathsf{T} - \mathbf{E}_k||_\text{F}^2 - \log \beta \rangle + \text{const} \\ &= (\frac{N}{2} mn - 1) \log \beta - \frac{\beta}{2} \sum_k \langle ||\mathbf{Y}_k - \mathbf{U_c} \mathbf{T}_k \mathbf{U_r}^\mathsf{T} - \mathbf{E}_k||_\text{F}^2 \rangle + \text{const} \end{aligned}$$

So:

$$q^*(\beta) = \text{Ga} \left(\frac{N}{2} mn, \frac{2}{\sum_k \langle ||\mathbf{Y}_k - \mathbf{U_c} \mathbf{T}_k \mathbf{U_r}^\mathsf{T} - \mathbf{E}_k||_\text{F}^2 \rangle} \right) \tag{6.19}$$

Update equation for $\boldsymbol{\gamma}_i$

$$\begin{aligned}\log q^*(\boldsymbol{\gamma}_i) &= \langle \sum_j \log p(u_{cj}|\boldsymbol{\gamma}_j) + \sum_j \log p(u_{rj}|\boldsymbol{\gamma}_j) + \sum_j \log p(\boldsymbol{\gamma}_j)\rangle + \text{const} \\ &= \langle \log p(\mathbf{u}_{ci}|\boldsymbol{\gamma}_i) + \log p(\mathbf{u}_{ri}|\boldsymbol{\gamma}_i) + \log p(\boldsymbol{\gamma}_i)\rangle + \text{const} \\ &= (a - 1 + \frac{m+n}{2}) \log \boldsymbol{\gamma}_i - \langle \frac{1}{b} + \frac{\mathbf{u}_{ci}^\mathsf{T}\mathbf{u}_{ci} + \mathbf{u}_{ri}^\mathsf{T}\mathbf{u}_{ri}}{2} \rangle \boldsymbol{\gamma}_i + \text{const} \\ &= (a - 1 + \frac{m+n}{2}) \log \boldsymbol{\gamma}_i - \left(\frac{1}{b} + \frac{\langle \mathbf{u}_{ci}^\mathsf{T}\mathbf{u}_{ci} \rangle + \langle \mathbf{u}_{ri}^\mathsf{T}\mathbf{u}_{ri} \rangle}{2} \right) \boldsymbol{\gamma}_i + \text{const}\end{aligned}$$

So:

$$q^*(\boldsymbol{\gamma}_i) = \text{Ga}\left(a + \frac{m+n}{2}, \frac{2b}{2 + b(\langle \mathbf{u}_{ci}^\mathsf{T}\mathbf{u}_{ci} \rangle + \langle \mathbf{u}_{ri}^\mathsf{T}\mathbf{u}_{ri} \rangle)} \right) \tag{6.20}$$

6.4 Required intermediary computations

In order to obtain closed-form expressions that we can implement algorithmically, we first need preliminary computations.

6.4.1 Expectation of $\mathbf{w}_{ni}^\mathsf{T}\mathbf{w}_{ni}$

The term $\langle \mathbf{w}_{ni}^\mathsf{T}\mathbf{w}_{ni} \rangle$ appears when computing the expectations of $\mathbf{u}_{ci}$ and $\mathbf{u}_{ri}$. We show how to compute them.

Lemma 6.4.1. *Let $\mathbf{x}$ a Gaussian vector with distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathbf{P}$ a matrix. Then $\mathbf{Px}$ has distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{P}\boldsymbol{\Sigma}\mathbf{P}^\mathsf{T})$.*

Recall $\mathbf{W}_n = \mathbf{U_r}\mathbf{T}_n^\mathsf{T}$, it follows that:

$$\mathbf{w}_{ni} = \mathbf{U_r}\mathbf{T}_{n(i,:)}$$

Where $\mathbf{T}_{n(i,:)}$ denotes the i^{th} row of $\mathbf{T}_n$. Our method is to extract the coefficients of $\mathbf{T}_{n(i,:)}$ from $\text{vec}(\mathbf{T}_n)$ which has a known distribution.

To do this, we observe that the elements of row i are at positions $i, i+r, i+2r, \ldots, i+(r-1)r$ in $\text{vec}(\mathbf{T}_n)$, and we define the row-extraction matrix $\mathbf{S}_i$ as follows.

Definition 6.4.1. We define the row-extraction matrix $\mathbf{S}_i$ as the matrix of $\mathbb{R}^{r \times r^2}$ that has ones at positions $(1,i), (2, i+r), \ldots, (k, i+(k-1)r), \ldots, (r, i+(r-1)r)$:

$$\mathbf{S}_i = \begin{pmatrix} 0 & \ldots & 1 & \ldots & \ldots & 0 \\ \vdots & & & & & \vdots \\ 0 & \ldots & \ldots & \ldots & 1 & 0 \end{pmatrix}$$

And we have the following result:

Proposition 6.4.1. *By definition of $\mathbf{S}_i$,*

$$\mathbf{S}_i \text{vec}(\mathbf{T}_n) = \mathbf{T}_{n(i,:)}$$

And using Lemma 6.4.1, $\mathbf{T}_{n(i,:)}$ is a Normal vector with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$:

$$\boldsymbol{\mu} = \mathbf{S}_i \langle \mathrm{vec}(\mathbf{T}_n) \rangle$$
$$\boldsymbol{\Sigma} = \mathbf{S}_i \left(I_{r^2} + \langle \beta \rangle \langle {\mathbf{U_r}}^\mathsf{T} \mathbf{U_r} \rangle \otimes \langle {\mathbf{U_c}}^\mathsf{T} \mathbf{U_c} \rangle \right)^{-1} \mathbf{S}_i^\mathsf{T}$$

We have $\mathbf{w}_{ni}^\mathsf{T} \mathbf{w}_{ni} = \mathrm{tr}\left[\mathbf{w}_{ni} \mathbf{w}_{ni}^\mathsf{T}\right] = \mathrm{tr}\left[\mathbf{T}_{n(i,:)}^\mathsf{T} {\mathbf{U_r}}^\mathsf{T} \mathbf{U_r} \mathbf{T}_{n(i,:)}\right] = \mathrm{tr}\left[{\mathbf{U_r}}^\mathsf{T} \mathbf{U_r} \mathbf{T}_{n(i,:)} \mathbf{T}_{n(i,:)}^\mathsf{T}\right]$.
By linearity of the trace and of the expectation we therefore have:

$$\langle \mathrm{tr}\left[{\mathbf{U_r}}^\mathsf{T} \mathbf{U_r} \mathbf{T}_{n(i,:)} \mathbf{T}_{n(i,:)}^\mathsf{T}\right] \rangle = \mathrm{tr}\left[\langle {\mathbf{U_r}}^\mathsf{T} \mathbf{U_r} \rangle \langle \mathbf{T}_{n(i,:)} \mathbf{T}_{n(i,:)}^\mathsf{T} \rangle\right]$$

$\langle {\mathbf{U_r}}^\mathsf{T} \mathbf{U_r} \rangle$ is known and $\langle \mathbf{T}_{n(i,:)} \mathbf{T}_{n(i,:)}^\mathsf{T} \rangle = \boldsymbol{\Sigma} + \boldsymbol{\mu}\boldsymbol{\mu}^\mathsf{T}$.

6.4.2 Expectation of $||\mathbf{X}_n - \mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2$

By expanding $||\mathbf{X}_n - \mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2$ we get:

$$\begin{aligned} ||\mathbf{X}_n - \mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T} - \mathbf{E}_n||_\mathrm{F}^2 = \mathrm{tr}\left[\mathbf{X}_n^\mathsf{T} \mathbf{X}_n\right] + \mathrm{tr}\left[\mathbf{E}_n^\mathsf{T} \mathbf{E}_n\right] + \mathrm{tr}\left[(\mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T})^\mathsf{T} (\mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T})\right] \\ - 2\mathrm{tr}\left[\mathbf{X}_n^\mathsf{T} \mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T}\right] + 2\mathrm{tr}\left[\mathbf{E}_n^\mathsf{T} \mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T}\right] + 2\mathrm{tr}\left[\mathbf{E}_n^\mathsf{T} \mathbf{X}_n\right] \end{aligned}$$

$\mathbf{X}_n$ is not random and the terms which involve first order expectations are all trivial to compute since everything is separable. Only the second order expectations need more explanations.

Expectation of $\mathrm{tr}\left[\mathbf{E}_n^\mathsf{T} \mathbf{E}_n\right]$

We have immediately that:

$$\langle \mathrm{tr}\left[\mathbf{E}_n^\mathsf{T} \mathbf{E}_n\right] \rangle = \mathrm{tr}\left[\langle \mathbf{E}_n^\mathsf{T} \rangle \langle \mathbf{E}_n \rangle\right] + \sum_{i,j} (\sigma_{e_{ij}^n})^2$$

Expectation of $\mathrm{tr}\left[(\mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T})^\mathsf{T} (\mathbf{U_c} \mathbf{T}_n {\mathbf{U_r}}^\mathsf{T})\right]$

We first expand the transposition and use the cyclic property of the trace to get:

$$\mathrm{tr}\left[({\mathbf{U_c}}^\mathsf{T} \mathbf{U_c}) \mathbf{T}_n ({\mathbf{U_r}}^\mathsf{T} \mathbf{U_r}) \mathbf{T}_n^\mathsf{T}\right]$$

Then we can note that ${\mathbf{U_c}}^\mathsf{T} \mathbf{U_c}$ and ${\mathbf{U_r}}^\mathsf{T} \mathbf{U_r}$ are symmetric so they are equal to their transpose, and use the following identity:

$$\mathrm{tr}\left[\mathbf{A}^\mathsf{T} \mathbf{B} \mathbf{C} \mathbf{D}^\mathsf{T}\right] = \mathrm{vec}(\mathbf{A})^\mathsf{T} (\mathbf{D} \otimes \mathbf{B}) \mathrm{vec}(\mathbf{C})$$

So we have:

$$\mathrm{tr}\left[({\mathbf{U_c}}^\mathsf{T} \mathbf{U_c}) \mathbf{T}_n ({\mathbf{U_r}}^\mathsf{T} \mathbf{U_r}) \mathbf{T}_n^\mathsf{T}\right] = \mathrm{vec}({\mathbf{U_c}}^\mathsf{T} \mathbf{U_c})(\mathbf{T}_n \otimes \mathbf{T}_n) \mathrm{vec}({\mathbf{U_r}}^\mathsf{T} \mathbf{U_r})$$

And this allows us to group the terms in $\mathbf{T}_n$ and obtain a product of separable variables. We can then compute the expectation:

$$\langle \mathrm{tr}\left[({\mathbf{U_c}}^\mathsf{T} \mathbf{U_c}) \mathbf{T}_n ({\mathbf{U_r}}^\mathsf{T} \mathbf{U_r}) \mathbf{T}_n^\mathsf{T}\right] \rangle = \langle \mathrm{vec}({\mathbf{U_c}}^\mathsf{T} \mathbf{U_c}) \rangle (\mathbf{T}_n \otimes \mathbf{T}_n) \mathrm{vec}({\mathbf{U_r}}^\mathsf{T} \mathbf{U_r})$$

$$= \langle \mathrm{vec}(\mathbf{U_c}^\mathsf{T}\mathbf{U_c})\rangle\langle \mathbf{T}_n \otimes \mathbf{T}_n\rangle\langle \mathrm{vec}(\mathbf{U_r}^\mathsf{T}\mathbf{U_r})\rangle$$
$$= \mathrm{vec}(\langle \mathbf{U_c}^\mathsf{T}\mathbf{U_c}\rangle)\langle \mathbf{T}_n \otimes \mathbf{T}_n\rangle\mathrm{vec}(\langle \mathbf{U_r}^\mathsf{T}\mathbf{U_r}\rangle)$$

$\langle \mathbf{U_c}^\mathsf{T}\mathbf{U_c}\rangle$ and $\langle \mathbf{U_r}^\mathsf{T}\mathbf{U_r}\rangle$ are known results. The interesting part is computing the expectation of the Kronecker product.

By inspection of the Kronecker product matrix we see that we need to compute the expectations of cross-terms of the form $\langle t_{ij}t_{kl}\rangle = \langle t_{ij}\rangle\langle t_{kl}\rangle + \mathrm{cov}(t_{ij}, t_{kl})$ and that the matrix $\mathbf{T}_n \otimes \mathbf{T}_n$ is organised in blocks.

The covariances we need are actually easy to retrieve from the covariance matrix of $\mathrm{vec}(\mathbf{T}_n)$ which is known. As such, let us consider the block of $\mathbf{T}_n \otimes \mathbf{T}_n$ obtained by multiplying t_{ij} with $\mathbf{T}_n$. The covariances involved are the covariances of t_{ij} with all other elements t_{kl} including itself and corresponds to **the $i + rj$ column of the covariance matrix** of $\mathrm{vec}(\mathbf{T}_n)$. So we can express the expectation of this block as:

$$\langle t_{ij}\rangle\mathbf{T}_n = \langle t_{ij}\rangle\langle \mathbf{T}_n\rangle + \mathrm{cov}(t_{ij}, \mathbf{T}_n)$$

Where $\mathrm{cov}(t_{ij}, \mathbf{T}_n)$ is an $r \times r$ matrix obtained by **folding the corresponding column of the covariance matrix of $\mathrm{vec}(\mathbf{T}_n)$ back into matrix form**:

$$\mathrm{cov}(t_{ij}, \mathbf{T}_n) = \mathrm{fold}(\mathbf{\Sigma}^{\mathrm{vec}(\mathbf{T}_n)}_{:,i+rj})$$

And fold is the inverse of the vec operator.

This leads to the final conclusion that:

$$\langle \mathbf{T}_n \otimes \mathbf{T}_n\rangle = \langle \mathbf{T}_n\rangle \otimes \langle \mathbf{T}_n\rangle + \mathbb{V}$$

With $\mathbb{V}$ the block matrix created from the block covariances as explained above.

6.5 Performance

Our implementation of the Bayesian algorithm suffers from numerical instability that we haven't been able to diagnose yet. We leave this for future work.

Chapter 7

Conclusion

The recent advances in the field of compressed sensing have reshaped our view of signal processing and provide fertile ground for innovation that will have long-lasting impact on science and technology.

Tensors being natural representations for many types of data, the development of efficient methods for robust tensor factorisation is of great importance and has received increasing interest from the statistics and artificial intelligence communities. In this thesis, we made several contributions to the field with a particular focus on computer vision applications. We provided an extensive literature review of both optimisation-based and Bayesian methods and compiled the essentials of the theoretical knowledge that underlies them in our background chapter. We then presented novel two-dimensional decompositions for robust tensor factorisation, provided a thorough assessment of their performance and of their complexity, and discussed possible extensions to the large-scale setting. Lastly, we highlighted important connections between our models and robust instances of Kronecker-decomposable dictionary learning.

In Chapter 5, we provided an experimental evaluation of a set of current state of the art robust tensor factorisation algorithms chosen to represent the main approaches that have been investigated to date. We reported four quantitative image quality measures that assess different aspects of image reconstruction, and compared the information they provide with visual inspection of the results. We showed our two best models, *RPCA2D Fro* ℓ_1, and *RPCA2D Fro* ℓ_2 consistently outperformed or matched the current best algorithms on all benchmarks. We also showed that *RPCA2D GL* ℓ_1 was capable of outperforming the two aforementioned methods but was also less robust to noise, yielding an overall less consistent performance. The second method based on *structural* regularisation for rank reduction, *RPCA2D GL* ℓ_2, didn't perform well in most benchmarks. Based on our experiments, we conclude the *spectral* Frobenius penalty on $\mathcal{T}$ gave higher reconstruction quality than the Group Lasso penalty, and that sparsity in the core tensor improved both the recovery performance and the robustness to noise. We also remind the reader that providing prior information on the rank in the form of a close upper bound can improve both the performance and the speed of our methods.

Finally, we set the basis for a Bayesian treatment of our methods and derived a Variational Bayes algorithm for inference in the associated graphical model. Although this approach hasn't proved successful for now, we believe it deserves to be studied in more depth in future work.

Our work on optimisation-based algorithm is currently being prepared for publication in the near future. Our literature review, novel methods, and experimental assessment are being condensed into a journal article. The novel methods alone will be submitted to a suitable computer vision or signal processing conference.

7.1 Future work and extensions

Several extensions have been envisioned by the author, both for optimisation-based and Bayesian methods.

7.1.1 Sparse dictionary learning

We believe the connections between our methods and sparse dictionary learning are worth future investigations and can open new research opportunities.

7.1.2 Applicability to data compression

We mentioned in section 4.7.3 that promoting sparsity in the core tensor could yield exact numerical zeros in the core and in the bases, and we discussed in section 4.8.4 possible extensions to sparse numerical linear algebra for reduce time and memory complexity at sufficient sparsity levels. Moreover, we showed our sparse methods can represent synthetic and real-world data with very high accuracy, including an exact recovery of, not only the magnitude of the elements of the synthetic sparse component, but also the exact number of non-zero elements.

Putting everything together, our sparse algorithms coupled with a sparse linear algebra implementation have an interesting potential as data compression methods. With dense storage, could further improve their applicability by allowing for rectangular frontal slices for the core tensor $\mathcal{T}$; this would reduce the storage needed by making it possible to use two different upper bounds for the 1-rank and the 2-rank of $\mathcal{T}$.

7.1.3 Visual quality assessment by a human jury

We mentioned repeatedly in Chapter 5 that quantitative image quality measures have their own limitations and that we could not always rely on them to accurately compare the perceived quality of the reconstructed images. We therefore propose a different approach for future comparative experiments and suggest asking a human jury to rank the reconstructed images by perceived quality. The final ranking would be determined by averaging the rankings given by every member of the jury.

7.1.4 Orthogonal bases

One possible extension would be to allow for orthogonality constraints to be put on the bases; the author believes the framework of Riemannian optimisation and differential geometry would provide natural and powerful tools to tackle this problem efficiently by optimisation on matrix manifolds (see Absil et al. [2]). Most notably, the *natural gradient* on the Stiefel manifold (*i.e,* the manifold of orthogonal matrices) is an attractive and flexible way of taking into account orthogonality constraints. We refer to Panagakis et al. [75] for an example case for the natural gradient.

7.1.5 Non-visual data

We could investigate how the algorithms we developed in this thesis can be applied to non-visual data. Tensor methods have long been used in fields such as psychometrics, or chemometrics, so assessing the applicability of our methods to these kinds of data would be an interesting extension. We also believe our methods can be used for source separation in audio applications, or for other kinds of blind source separation.

7.1.6 Extension to other norms and missing values

The algorithm we developed for RPCA2D with Group Lasso can be extended to any non-smooth penalty; we implemented a variant which replaces the ℓ_1/ℓ_2 norms by nuclear norms but, due to the limited time frame of the present project, chose not to include it in this thesis. The author also suggests experimenting with an Elastic Net (Zou and Hastie [121]) penalty on $\mathcal{T}$ or on the bases; or a *sparse* Group Lasso on the bases. Most notably, the author suggests investigating the suitability of other Schatten-p norms on the bases, and ℓ_p or ℓ_p/ℓ_q norms on the core. We refer to [77] for a recent development regarding matrix Robust PCA. Another extension we mentioned in Chapter 4 would be to use structured sparsity on the sparse outliers term.

Finally, we believe our methods can be extended to handle missing values and therefore be used for tensor completion.

7.1.7 Bayesian model with time correlation

On the Bayesian side, authors recently introduced tensor methods for tensor completion in collaborative filtering that model time correlation between user ratings via a Markov chain (see Xiong et al. [108]) - more precisely a first-order linear dynamical system. We suggest adapting this technique to video data to take into account time correlation between video frames. Our idea is to make the model robust to outliers and to extend it to an n^{th} order auto-regressive model. The order of the model being unknown, we suggest that a Bayesian treatment could be of interest. Ehlers and Brooks [33], Enciso-Mora et al. [34] discuss order-selection for ARMA and ARIMA models in a Bayesian setting via reversible-jump MCMC.

7.1.8 Theoretical analysis

Finally, a more thorough analysis of the convergence and optimality properties of our algorithms would be of high interest. Several authors, including Candès et al. [21], Lu et al. [66] give optimal choices for the penalty weight on the outliers based on restricted isometry properties and it would be interesting to determine whether such criterion exists for our methods. Goldfarb and Qin [40], Yang et al. [110] provide convergence analyses that could serve as the basis for the convergence analysis of our models.

Appendix A

Complexity Analysis

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n \times m \times N}$. We first present the complexity analysis of the linear-algebra operations that form the building blocks of our algorithms:

- **Tensor soft-shrinkage operator** $\mathcal{S}_\lambda$: this operator applies the same operation to each element of $\boldsymbol{\mathcal{X}}$, so requires $O(Nmn)$ flops. The operations can be done in-place, so an ideal implementation doesn't require additional storage, but in the worst case we would need $O(Nmn)$ space. In Algorithm 1 and its variants, we never need to keep the un-shrunk arrays.

- **Matrix transposition**: We made the assumption that the linear-algebra libraries used by scientific computing packages were able to handle transposition by accessing the array to transpose in the right order. This assumption is realistic for packages relying on BLAS and LAPACK (*i.e,* most existing software). So no additional complexity is added.

- **Matrix multiplication**: Let $\mathbf{A} \in \mathbb{R}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times m}$, the time complexity of computing $\mathbf{AB}$ is assumed to be $O(mnk)$ for simplicity. The space complexity is $O(mn)$.

- **Solving a linear system**: We consider the case we encounter in our methods, that is right-dividing by an $r \times r$ matrix. Since software such as Matlab employ a variety of numerical solvers depending on the structure of the matrix, we will consider the cost to be $O(r^3)$. This is a reasonable assumption considering the matrices involved in our methods are symmetric, likely to be positive-definite, so the systems are likely to be solvable by Cholesky decomposition. Most algorithms for Cholesky factorisation are of cubic complexity, and the cost of Gaussian elimination is also cubic but solving triangular systems is quadratic. In the worst case, matrix inversion also has cubic complexity. Regarding the storage, we assume that we will need additional quadratic space: $O(mn)$ or $O(r^2)$ for square matrices.

- **Solving the Stein equation**: Both the Hessenberg-Schur and the Bartel-Stewarts algorithms require a number of floating-point operations cubic in the dimensions of the matrices and quadratic storage [41]. So computing $\mathbf{T}_n$ by solving the Stein equation requires $O(r^3)$ time and $O(r^2)$ memory.

- **Tensor scaling**: Computing $\alpha\boldsymbol{\mathcal{X}}$ from $\boldsymbol{\mathcal{X}}$ requires $O(Nmn)$ floating-point operations. In some cases it is acceptable to alter the array in-place, yielding constant-space requirements, but in general we cannot afford overwriting the array and need an additional $O(Nmn)$ space.

- **Tensor addition and subtraction**: Computing $\boldsymbol{\mathcal{X}} + \boldsymbol{\mathcal{Y}}$ requires as many sums as there are elements, so $O(Nmn)$ operations are needed. Modern computer architectures are able to

subtract floating point numbers directly[1] so the cost is the same. Addition and subtraction can be performed in-place, and if not, we would need $O(Nmn)$ storage for the result.

- **Singular Value Decomposition**: We only compute SVDs in the initialisation procedure, so the cost of an SVD doesn't impact the *per-iteration* complexity but needs to be taken into account in our assessment of the complexity of the whole algorithms. Let $\mathbf{X} \in \mathbb{R}^{m \times n}$, computing the SVD requires [24] $O(mn \min(m,n) + (\min(m,n))^3)$ floating-point operations. The resulting triple $(\mathbf{U}, \mathbf{S}, \mathbf{V})$ must be stored in addition to $\mathbf{X}$ and requires $m \times \min(m,n) + \min(m,n)^2 + n \times \min(m,n)$ storage.

A.1 Analysis of the base algorithm

We can now outline the computation of the cost per iteration of Algorithm 1, which we reproduce below for the reader's convenience:

Algorithm 1 Non-orthogonal 2D RPCA

1: **procedure** NO2DRPCA($\boldsymbol{\mathcal{X}}, r, \lambda, \alpha_t, \alpha_c, \alpha_r$)
2: $\mathbf{U_c}^0, \mathbf{U_r}^0, \boldsymbol{\mathcal{E}}^0, \boldsymbol{\mathcal{T}}^0 \leftarrow$ INITIALIZE($\boldsymbol{\mathcal{X}}$)
3: **while** not converged **do**
4: $\boldsymbol{\mathcal{E}}^{t+1} \leftarrow \mathcal{S}_{\lambda/\mu^t}(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{T}}^t \times_1 \mathbf{U}_c^t \times_2 \mathbf{U}_r^t + \frac{1}{\mu^t}\boldsymbol{\Lambda}^t)$
5: $\tilde{\boldsymbol{\mathcal{X}}}^{t+1} \leftarrow \boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{E}}^{t+1}$
6: $\mathbf{U}_c^{t+1} \leftarrow (\sum_n (\mu^t \tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t)\mathbf{U}_r^t(\mathbf{T}_n^t)^\mathsf{T}) \,/\, (\alpha_c \mathbf{I} + \mu^t \sum_n \mathbf{T}_n^t(\mathbf{U}_r^t)^\mathsf{T}\mathbf{U}_r^t(\mathbf{T}_n^t)^\mathsf{T})$
7: $\mathbf{U}_r^{t+1} \leftarrow (\sum_n (\mu^t \tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t)^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{T}_n^t) \,/\, (\alpha_r \mathbf{I} + \mu^t \sum_n (\mathbf{T}_n^t)^\mathsf{T}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}\mathbf{T}_n^t)$
8: **for all** n **do**
9: $\mathbf{T}_n^{t+1} \leftarrow$ STEIN($-\frac{\mu^t}{\alpha_t}(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1},\ (\mathbf{U}_r^{t+1})^\mathsf{T}\mathbf{U}_r^{t+1},\ \frac{1}{\alpha_t}(\mathbf{U}_c^{t+1})^\mathsf{T}(\boldsymbol{\Lambda}_n^t + \mu^t \tilde{\mathbf{X}}_n^{t+1})\mathbf{U}_r^{t+1}$)
10: **end for**
11: $\boldsymbol{\Lambda}^{t+1} \leftarrow \boldsymbol{\Lambda}^t + \mu^t(\tilde{\boldsymbol{\mathcal{X}}}^{t+1} - \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1})$
12: $\mu^{t+1} \leftarrow \rho\mu^t$ ▷ ρ is a parameter that can be tuned
13: **end while**
14: **return** $\mathbf{U_c}, \mathbf{U_r}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{E}}$ ▷ The low-rank tensor is $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{T}}^{t+1} \times_1 \mathbf{U}_c^{t+1} \times_2 \mathbf{U}_r^{t+1}$
15: **end procedure**

We only need to compute $\mu^t \tilde{\mathbf{X}}_n^{t+1} + \boldsymbol{\Lambda}_n^t$ once for each n for lines 6, 7, and 9. Similarly, once $\mathbf{U}_c^{t+1}$ has been computed, we can pre-compute $(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}$ for lines 7 and 9. In general and in the worst case, we assume storage cannot be re-used unless clearly stated otherwise. Since the order of mode-n products on different modes doesn't matter, and matrix multiplication is associative, we always choose the least costly order to perform the multiplications. This translates by a time and space complexity that depends on $\min(m,n)$.

Line 4: Updating $\boldsymbol{\mathcal{E}}$ requires $O(N(mnr + mn + \min(m,n)r^2))$ operations for $2N$ matrix-matrix multiplications, one scaling operation, one tensor sum, one tensor subtraction, and then the shrinkage. The space complexity is $O(N(mn + \min(m,n)r))$.

Line 5: Computing $\tilde{\boldsymbol{\mathcal{X}}}$ is one subtraction so $O(Nmn)$ time and space. We can now compute $\boldsymbol{\mathcal{W}}^{t+1} = \mu\tilde{\boldsymbol{\mathcal{X}}}^{t+1} + \boldsymbol{\Lambda}^{t+1}$ for one tensor scaling and one tensor sum, this gives a cost in flops of $O(Nmn)$ and requires $O(Nmn)$ storage.

[1] On x86 machines, see the `subsd` instruction

Line 6 and 7: Let us split the computation in three parts; first the element on the left of the slash operator (LHS), then the one on the right (RHS), and finally the cost of solving the system.

Having computed $\boldsymbol{\mathcal{W}}$, the LHS is the sum of N matrix-matrix products $\mathbf{W}_n\mathbf{U_r}\mathbf{T}_n^\mathsf{T}$. Each product has a cost of $O(nmr + \min(m,n)r^2)$ in time and $O(nr + \min(m,n)r)$ in space, so the total cost is $O(N(nmr + \min(m,n)r^2)$ time and $O(N(nr + \min(m,n)r))$ space assuming we compute each product separately before the reduction. We then need to compute $N-1$ sums to reduce the N matrices to their sum, which gives a cost of $O((N-1)nr)$ floating point operations and requires $O(nr)$ storage for accumulating the results. The space complexity can be brought down to $O(nr)$ since we can initialise an accumulator by computing the first product in the sum, and compute and add one product at a time; note however that this prevents parallelising the computations. The total cost is:

$$O(N(nmr + \min(m,n)r^2) + (N-1)nr) \text{ time and } O(N(nr + \min(m,n)r) + nr) \text{ space}$$

Or $O(nr + \min(m,n)r)$ space with sequential computation of the sum.

Computing the RHS also involves reducing N matrix-matrix products to their sum. We first compute $(\mathbf{U}_r^t)^\mathsf{T}\mathbf{U}_r^t$ for $O(mr^2)$ operations and $O(r^2)$ memory since this factor appears in all the products in the sum. We then need N matrix-matrix products $\mathbf{T}_n(\mathbf{U_r}^\mathsf{T}\mathbf{U_r})\mathbf{T}_n^\mathsf{T}$ of total cost $O(Nr^3)$ time and $O(Nr^2)$ space. Reducing the matrices to their sum then requires $O((N-1)r^2)$ flops and $O(r^2)$ space . Note again that computing the sum one product at a time only requires $O(r^2)$ storage. Taking into account the scaling and the addition of the $r \times r$ identity matrix, we find a total cost of:

$$O(Nr^3 + (N-1)r^2 + mr^2 + r^2) \text{ time and } O((N+1)r^2) \text{ space}$$

Or $O(r^2)$ space with sequential computation of the sum.

Finally, solving the system requires $O(r^3)$ flops and $O(r^2)$ space. Having updated $\mathbf{U_c}$, we can compute $(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}$ at cost $O(nr^2)$ time and $O(r^2)$ storage.

By symmetry, updating $\mathbf{U_r}$ requires a total of:

$$\begin{aligned} O(N(nmr + \min(m,n)r^2 + r^3) + (N-1)(mr + r^2) + nr^2 + r^3 + r^2) \quad &\text{time} \\ O(N(mr + \min(m,n)r) + (N+1)r^2 + mr) \text{ or } O(mr + r^2) \quad &\text{space} \end{aligned}$$

Lines 8 to 10: We first compute the factors common to all updates: $(\mathbf{U}_c^{t+1})^\mathsf{T}\mathbf{U}_c^{t+1}$ is known, scaling the matrix requires $O(r^2)$ time and space. $(\mathbf{U}_r^{t+1})^\mathsf{T}\mathbf{U}_r^{t+1}$ must be computed for $O(mr^2)$ time and $O(r^2)$ space.

For each of the N update we compute $\frac{1}{\alpha_t}\mathbf{U_c}^\mathsf{T}\mathbf{W}_n\mathbf{U_r}$, this has cost $O(mnr + mr^2 + r^2)$ or $O(mnr + nr^2 + r^2)$ flops depending on the order in which the products are done, and requires $O(mr + r^2)$ or $O(nr + r^2)$ storage - which can be shared. The Hessenberg-Schur method has costs $O(r^3)$ time and $O(r^2)$ space.

This gives a total cost of:

$$\begin{aligned} O(N(mnr + \min(m,n)r^2 + r^2) + mr^2 + r^3 + r^2) \quad &\text{time} \\ O(\min(m,n)r + r^2) \quad &\text{space} \end{aligned}$$

Line 11: Taking into account the cost of two mod-n products, one tensor subtraction, one tensor scaling, and one tensor addition, we get a total cost of:

$$\begin{aligned} O(N(mnr + mn + \min(m,n)r^2)) \quad &\text{time} \\ O(N(mn + \min(m,n)r)) \quad &\text{space} \end{aligned}$$

Summary Based on the previous discussion, Algoritm 1 has complexity:

$$O(N(mnr + (m+n)r + mn + \min(m,n)r^2 + r^3 + r^2) + (m+n)r^2 + r^3 + r^2) \quad \text{time}$$
$$O(N(mn + (m+n)r + r^2) + (m+n)r + r^2) \text{ or } O(Nmn + (m+n)r + r^2) \quad \text{space}$$

Hence the reported complexity.

A.2 Analysis of the variants

Now that the complexity of the base algorithm has been determined, we can analyse what additional cost each of the variants propose.

A.2.1 Sparse core tensor

The additional complexity of this variant comes from the use of splitting and from applying soft-shrinkage to $\mathcal{K}$. Taking into account the changes in the update of $\mathcal{K}$ compared to that of $\mathcal{T}$ in Algorithm 1, the new Lagrange multipliers and their updates, and the computation of $\mathcal{T}$, we find the additional cost to be:

$$O(Nr^2) \text{ time and space}$$

A.2.2 Group Lasso penalties

In this variant, the added cost comes from splitting and from the proximal mapping of the ℓ_1/ℓ_2 norm. Updating the additional Lagrange multipliers as well as accounting for the slightly different updates of $\mathbf{A}$ and $\mathbf{B}$ compared to these of $\mathbf{U_c}$ and $\mathbf{U_r}$ in Algorithm 1 has cost $O(nr)$ time and space for $\mathbf{A}/\mathbf{U_c}$ and $O(mr)$ time and space for $\mathbf{B}/\mathbf{U_r}$.

Computing the proximal operator involves computing $\max(\mathbf{x} - \lambda\frac{\mathbf{x}}{||\mathbf{x}||_2}, 0)$ where x is a column of $\mathbf{U_c}$ or $\mathbf{U_r}$. There are r columns of dimension n (or m), and the time and space cost is $O(n)$ (or $O(m)$) per column. Therefore the total additional complexity for this variant is:

$$O(nr + mr) \text{ time and space}$$

A.2.3 Robust estimation of the mean

Recall from equation 4.15 that the estimate of M at iteration $t+1$ is given by:

$$\hat{\mathbf{M}}^t = \frac{1}{N}\sum_n (\mathbf{X}_n - \mathbf{E}_n^t) \tag{A.1}$$

The computation involves N differences and $N-1$ sums as well as one final scaling, so the time complexity is:

$$O(Nmn + (N-1)mn + mn) = O(2Nmn) = O(Nmn)$$

The storage required is $O(mn)$ for the result, the scaling (possibly in place), and accumulating the sum. Similarly to our discussion on the update of $\mathbf{U_c}$ in the base algorithm, the terms within the sum can either be computed one at a time or all beforehand. The space complexity is therefore either $O(mn)$ or $O(Nmn)$.

Appendix B

Best parameters per algorithm

B.1 Facade

Algorithm	Param(s)	Best PSNR	Best SSIM	Best FSIMc	Best SAM
Tensor RPCA (Papamakarios et. al.)	λ	0.032	0.023	0.032	0.023
Tensor BRPCA (Papamakarios et. al.)	λ	0.067	0.067	0.067	0.067
Tensor RHOSVD (Papamakarios et. al.)	λ	0.023	0.023	0.032	0.023
Tensor RCPD (Papamakarios et. al.)	λ	0.0018	0.067	0.097	0.0076
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.023	0.023	0.011	0.011
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.023	0.023	0.023	0.023
Tensor RPCA (CVPR 2016)	λ	0.042	0.035	0.042	0.035
BRTF	IV	0.018	0.0089	0.0089	0.001
HORPCA-S	λ	0.09	0.077	0.09	0.077
HORPCA-S TC	λ	0.11	0.09	0.11	0.09
Tensor RPCA (CVPR 2014)	λ	0.016	0.012	0.016	0.0087
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0006 0.01	0.00041 0.01	0.0018 0.1	0.00041 0.00041
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.00029 1	0.00029 1	0.00029 0.1	0.0002 0.0002
Welsh ST	(σ, α)	0.29 0.077	0.29 0.077	0.29 0.077	0.24 0.24
Cauchy ST	(σ, α)	0.09 0.026	0.074 0.026	0.09 0.026	0.074 0.074
NC TRPCA	threshold rank	1e+03 77	1e+03 77	1e+03 77	1e+03 1e+03

Table B.1: Best parameters on Facade - 10% Salt & Pepper noise

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIMc**	**Best SAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.023	0.023	0.023	0.016
Tensor BRPCA (Papamakarios et. al.)	λ	0.047	0.047	0.047	0.032
Tensor RHOSVD (Papamakarios et. al.)	λ	0.016	0.016	0.023	0.016
Tensor RCPD (Papamakarios et. al.)	λ	0.0037	0.097	0.2	0.0012
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.011	0.0076	0.011	0.0076
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.032	0.032	0.032	0.016
Tensor RPCA (CVPR 2016)	λ	0.027	0.024	0.027	0.02
BRTF	IV	0.7	1.4	1.4	0.34
HORPCA-S	λ	0.065	0.056	0.065	0.048
HORPCA-S TC	λ	0.077	0.065	0.077	0.065
Tensor RPCA (CVPR 2014)	λ	0.012	0.0087	0.012	0.0087
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0006 0.01	0.0006 0.01	0.0006 0.01	0.0006 0.0006
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.00041 1	0.00029 0.1	0.00041 0.1	0.0002 0.0002
Welsh ST	(σ, α)	0.35 0.15	0.35 0.15	0.35 0.15	0.35 0.35
Cauchy ST	(σ, α)	0.061 0.026	0.061 0.026	0.061 0.026	0.061 0.061
NC TRPCA	threshold rank	1e+02 5e+02	1e+02 5e+02	1e+03 25	1e+03 1e+03

Table B.2: Best parameters on Facade - 30% Salt & Pepper noise

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIMc**	**Best SAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.016	0.016	0.016	0.016
Tensor BRPCA (Papamakarios et. al.)	λ	0.032	0.032	0.047	0.032
Tensor RHOSVD (Papamakarios et. al.)	λ	0.011	0.011	0.016	0.011
Tensor RCPD (Papamakarios et. al.)	λ	0.047	0.097	0.097	0.14
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.0025	0.0018	0.0025	0.0018
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.011	0.011	0.011	0.0076
Tensor RPCA (CVPR 2016)	λ	0.016	0.016	0.02	0.016
BRTF	IV	0.34	1.4	1.4	0.34
HORPCA-S	λ	0.048	0.048	0.056	0.048
HORPCA-S TC	λ	0.056	0.056	0.065	0.056
Tensor RPCA (CVPR 2014)	λ	0.012	0.012	0.012	0.012
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0006 1	0.00041 1	0.00041 1	0.0006 0.0006
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.023 0.001	0.023 0.0001	0.0006 1	0.016 0.016
Welsh ST	(σ, α)	0.62 0.51	0.51 0.38	0.51 0.36	0.51 0.51
Cauchy ST	(σ, α)	0.16 0.13	0.16 0.13	0.19 0.15	0.16 0.16
NC TRPCA	threshold rank	1e+02 3.9e+02	1e+02 3.4e+02	1e+02 3.9e+02	1e+02 1e+02

Table B.3: Best parameters on Facade - 60% Salt & Pepper noise

B.2 Yale-B - Salt & Pepper noise

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.016	0.011	0.011	0.016
Tensor BRPCA (Papamakarios et. al.)	λ	0.032	0.023	0.032	0.032
Tensor RHOSVD (Papamakarios et. al.)	λ	0.016	0.011	0.011	0.016
Tensor RCPD (Papamakarios et. al.)	λ	0.0037	0.0037	0.032	0.0025
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.011	0.023	0.023	0.023
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.0025	0.0018	0.0018	0.0025
Tensor RPCA (CVPR 2016)	λ	0.014	0.014	0.013	0.014
BRTF	IV	0.34	0.7	0.34	0.34
HORPCA-S	λ	0.041	0.03	0.035	0.041
HORPCA-S TC	λ	0.049	0.041	0.041	0.049
Tensor RPCA (CVPR 2014)	λ	0.083	0.079	0.083	0.083
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0037 0.1	0.0053 1	0.0076 1	0.0053 0.0053
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.0012 0.0001	0.0012 0.0001	0.0012 0.0001	0.0012 0.0012
Welsh ST	(σ, α)	0.26 0.21	0.26 0.21	0.26 0.18	0.26 0.26
Cauchy ST	(σ, α)	0.05 0.051	0.05 0.051	0.05 0.051	0.05 0.05
NC TRPCA	threshold rank	1e+03 95	1e+03 95	1e+03 1.1e+02	1e+03 1e+03

Table B.4: Best parameters on Yale-B - 10% Salt & Pepper noise

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.011	0.0076	0.011	0.011
Tensor BRPCA (Papamakarios et. al.)	λ	0.023	0.016	0.016	0.023
Tensor RHOSVD (Papamakarios et. al.)	λ	0.011	0.0076	0.0076	0.0076
Tensor RCPD (Papamakarios et. al.)	λ	0.011	0.011	0.016	0.011
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.023	0.016	0.023	0.016
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.0025	0.0025	0.0025	0.0025
Tensor RPCA (CVPR 2016)	λ	0.01	0.0091	0.0091	0.01
BRTF	IV	0.16	0.16	0.16	0.16
HORPCA-S	λ	0.03	0.026	0.026	0.03
HORPCA-S TC	λ	0.035	0.03	0.03	0.035
Tensor RPCA (CVPR 2014)	λ	0.11	0.1	0.11	0.11
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.016 0.1	0.016 0.1	0.016 0.1	0.016 0.016
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.00041 1	0.0002 1	0.0002 1	0.00041 0.00041
Welsh ST	(σ, α)	0.39 0.54	0.39 0.54	0.26 0.31	0.39 0.39
Cauchy ST	(σ, α)	0.075 0.1	0.075 0.1	0.05 0.077	0.075 0.075
NC TRPCA	threshold rank	1e+03 1.6e+02	1e+03 79	1e+03 79	1e+04 1e+04

Table B.5: Best parameters on Yale-B - 30% Salt & Pepper noise

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.0076	0.0076	0.0076	0.0076
Tensor BRPCA (Papamakarios et. al.)	λ	0.016	0.011	0.016	0.016
Tensor RHOSVD (Papamakarios et. al.)	λ	0.0076	0.0053	0.0076	0.0076
Tensor RCPD (Papamakarios et. al.)	λ	0.032	0.032	0.023	0.032
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.011	0.011	0.011	0.011
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.0025	0.0025	0.0025	0.0025
Tensor RPCA (CVPR 2016)	λ	0.0068	0.0061	0.0061	0.0068
BRTF	IV	0.001	0.001	0.001	0.018
HORPCA-S	λ	0.026	0.022	0.022	0.026
HORPCA-S TC	λ	0.03	0.026	0.03	0.03
Tensor RPCA (CVPR 2014)	λ	0.14	0.12	0.11	0.14
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0037 0.1	0.0002 1	0.00029 1	0.0037 0.0037
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.00041 0.0001	0.0002 1	0.0002 1	0.00029 0.00029
Welsh ST	(σ, α)	0.39 0.64	0.39 0.9	0.39 0.69	0.39 0.39
Cauchy ST	(σ, α)	0.17 0.41	0.17 0.44	0.17 0.44	0.26 0.26
NC TRPCA	threshold rank	1e+04 47	1e+03 63	1e+03 63	1e+04 1e+04

Table B.6: Best parameters on Yale-B - 60% Salt & Pepper noise

B.3 Yale-B - Patch corruption

Algorithm	Param(s)	Best PSNR	Best SSIM	Best FSIM	Best MSAM
Tensor RPCA (Papamakarios et. al.)	λ	0.023	0.018	0.018	0.018
Tensor BRPCA (Papamakarios et. al.)	λ	0.048	0.038	0.048	0.048
Tensor RHOSVD (Papamakarios et. al.)	λ	0.018	0.015	0.018	0.018
Tensor RCPD (Papamakarios et. al.)	λ	0.0071	0.0071	0.011	0.0017
RPCA2D Fro ℓ_1 w/ mean (Bahri et. al.)	λ	0.023	0.023	0.023	0.015
RPCA2D Fro ℓ_2 w/ mean (Bahri et. al.)	λ	0.0044	0.0044	0.0056	0.0044
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.009	0.009	0.015	0.009
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.0035	0.0035	0.0044	0.0035
Tensor RPCA (CVPR 2016)	λ	0.012	0.011	0.015	0.012
BRTF	IV	0.045	0.49	0.3	0.045
HORPCA-S	λ	0.057	0.041	0.057	0.057
HORPCA-S TC	λ	0.03	0.03	0.057	0.03
Tensor RPCA (CVPR 2014)	λ	0.095	0.12	0.095	0.1
RPCA2D GL ℓ_1 w/ mean (Bahri et. al.)	(λ, α)	0.016 1	0.016 1	0.016 1	0.016 0.016
RPCA2D GL ℓ_2 w/ mean (Bahri et. al.)	(λ, α)	0.0012 0.1	0.00086 0.01	0.0018 1	0.0012 0.0012
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.011 0.1	0.011 0.1	0.011 0.1	0.0076 0.0076
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.0012 0.01	0.0012 0.01	0.0018 1	0.0012 0.0012
Welsh ST	(σ, α)	0.58 0.38	0.39 0	0.26 0.026	0.58 0.58
Cauchy ST	(σ, α)	0.11 0.077	0.05 0	0.05 0.026	0.05 0.05
NC TRPCA	threshold rank	1e+04 1.3e+02	1e+03 1.3e+02	1e+03 1.1e+02	1e+03 1e+03

Table B.7: Best parameters on Yale-B - Patch of max size 50px

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.015	0.011	0.015	0.015
Tensor BRPCA (Papamakarios et. al.)	λ	0.038	0.023	0.03	0.03
Tensor RHOSVD (Papamakarios et. al.)	λ	0.015	0.011	0.011	0.011
Tensor RCPD (Papamakarios et. al.)	λ	0.0017	0.0017	0.0035	0.0017
RPCA2D Fro ℓ_1 w/ mean (Bahri et. al.)	λ	0.009	0.009	0.011	0.009
RPCA2D Fro ℓ_2 w/ mean (Bahri et. al.)	λ	0.0035	0.0035	0.0035	0.0035
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.009	0.011	0.011	0.009
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.0035	0.0027	0.0035	0.0027
Tensor RPCA (CVPR 2016)	λ	0.01	0.01	0.011	0.01
BRTF	IV	0.073	0.49	0.3	0.045
HORPCA-S	λ	0.041	0.03	0.041	0.041
HORPCA-S TC	λ	0.03	0.03	0.03	0.022
Tensor RPCA (CVPR 2014)	λ	0.085	0.15	0.15	0.085
RPCA2D GL ℓ_1 w/ mean (Bahri et. al.)	(λ, α)	0.0025 0.0001	0.047 0.0001	0.047 0.0001	0.047 0.047
RPCA2D GL ℓ_2 w/ mean (Bahri et. al.)	(λ, α)	0.00086 1	0.00086 1	0.0012 1	0.00086 0.00086
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0025 0.01	0.14 0.0001	0.14 0.0001	0.14 0.14
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.00086 1	0.00086 1	0.0012 0.01	0.00086 0.00086
Welsh ST	(σ, α)	0.58 0.95	0.39 0	0.26 0.051	0.58 0.58
Cauchy ST	(σ, α)	0.075 0.051	0.05 0	0.05 0.026	0.26 0.26
NC TRPCA	threshold rank	1e+04 47	1e+04 47	1e+04 1.4e+02	1e+03 1e+03

Table B.8: Best parameters on Yale-B - Patch of max size 100px

Algorithm	**Param(s)**	**Best PSNR**	**Best SSIM**	**Best FSIM**	**Best MSAM**
Tensor RPCA (Papamakarios et. al.)	λ	0.011	0.009	0.011	0.011
Tensor BRPCA (Papamakarios et. al.)	λ	0.023	0.018	0.023	0.023
Tensor RHOSVD (Papamakarios et. al.)	λ	0.011	0.009	0.009	0.009
Tensor RCPD (Papamakarios et. al.)	λ	0.0017	0.0017	0.0022	0.0017
RPCA2D Fro ℓ_1 w/ mean (Bahri et. al.)	λ	0.0044	0.0044	0.0044	0.0044
RPCA2D Fro ℓ_2 w/ mean (Bahri et. al.)	λ	0.0027	0.0027	0.0027	0.0027
RPCA2D Fro ℓ_1 (Bahri et. al.)	λ	0.0044	0.0056	0.0056	0.0056
RPCA2D Fro ℓ_2 (Bahri et. al.)	λ	0.0027	0.0022	0.0027	0.0027
Tensor RPCA (CVPR 2016)	λ	0.008	0.008	0.0085	0.008
BRTF	IV	0.0042	0.49	0.12	0.045
HORPCA-S	λ	0.03	0.03	0.03	0.03
HORPCA-S TC	λ	0.016	0.022	0.022	0.016
Tensor RPCA (CVPR 2014)	λ	0.1	0.15	0.15	0.09
RPCA2D GL ℓ_1 w/ mean (Bahri et. al.)	(λ, α)	0.0037 1	0.047 1	0.047 1	0.0025 0.0025
RPCA2D GL ℓ_2 w/ mean (Bahri et. al.)	(λ, α)	0.00086 0.0001	0.00086 0.1	0.00086 0.0001	0.0006 0.0006
RPCA2D GL ℓ_1 (Bahri et. al.)	(λ, α)	0.0018 1	0.097 0.0001	0.067 0.1	0.0018 0.0018
RPCA2D GL ℓ_2 (Bahri et. al.)	(λ, α)	0.00086 1	0.0012 1	0.00086 1	0.00086 0.00086
Welsh ST	(σ, α)	0.58 1	0.39 0	0.26 0.026	0.58 0.58
Cauchy ST	(σ, α)	0.11 0.13	0.075 0.077	0.05 0.026	0.39 0.39
NC TRPCA	threshold rank	1e+03 15	1e+03 15	1e+04 79	1e+03 1e+03

Table B.9: Best parameters on Yale-B - Patch of max size 160px

Appendix C

Parameter tuning: experimental plots

C.1 Facade

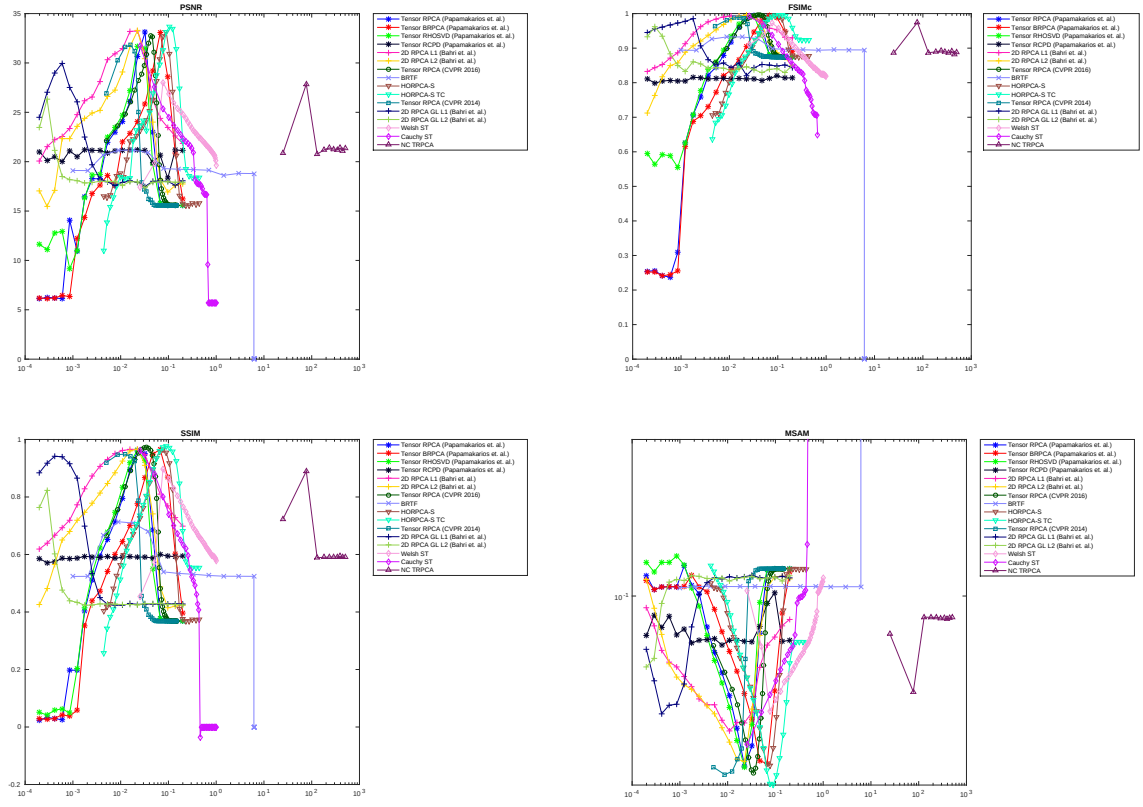

Figure C.1: Evolution of the four measures on the Facade benchmark at 10% salt & pepper noise

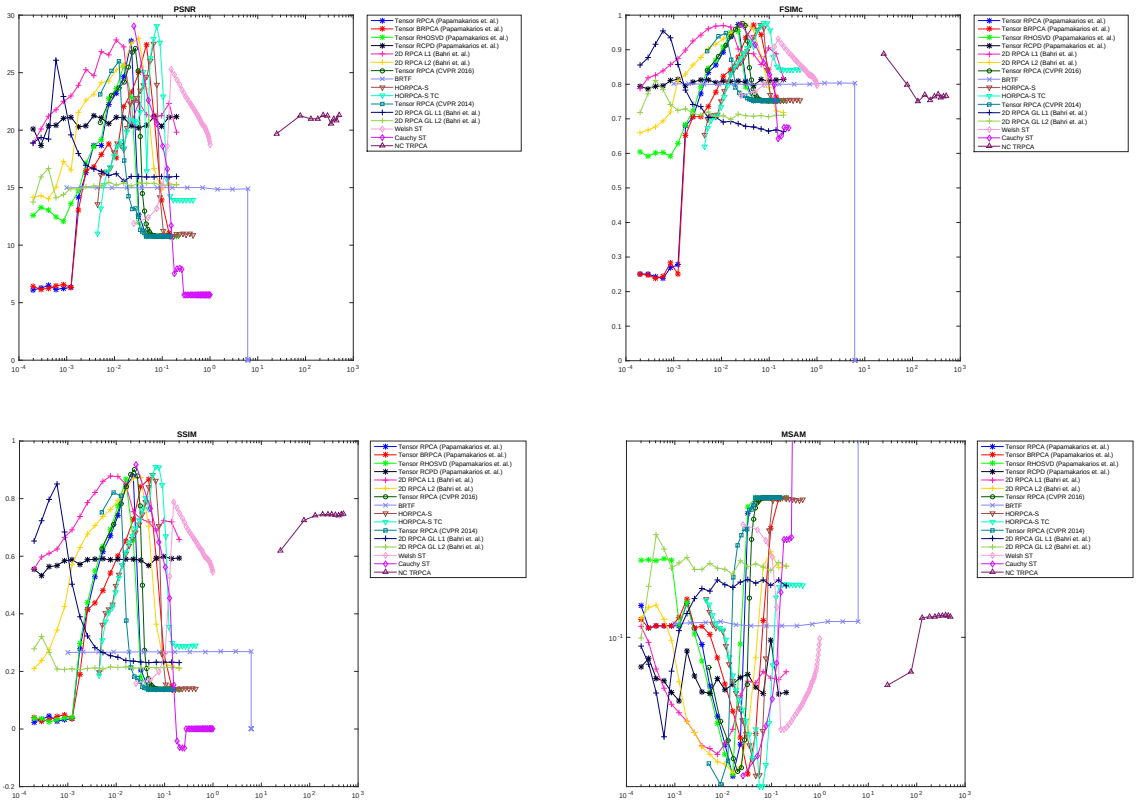

Figure C.2: Evolution of the four measures on the Facade benchmark at 30% salt & pepper noise

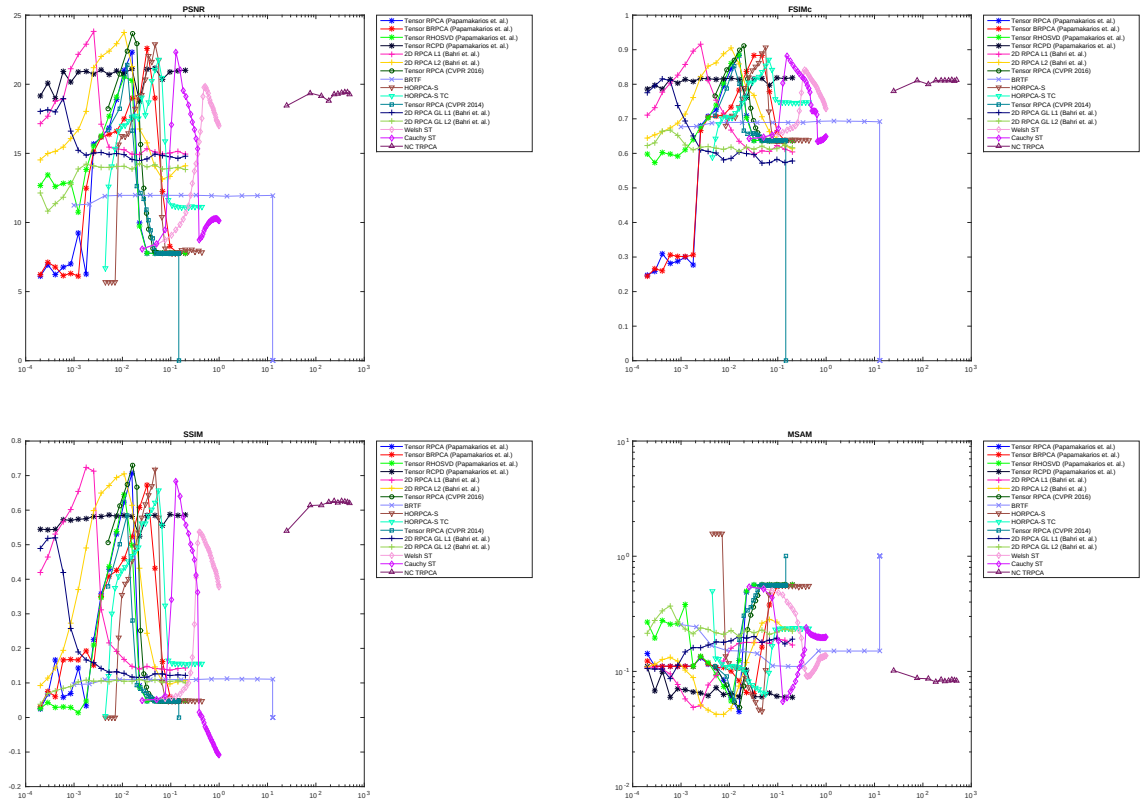

Figure C.3: Evolution of the four measures on the Facade benchmark at 60% salt & pepper noise

C.2 Yale-B - Salt & Pepper noise

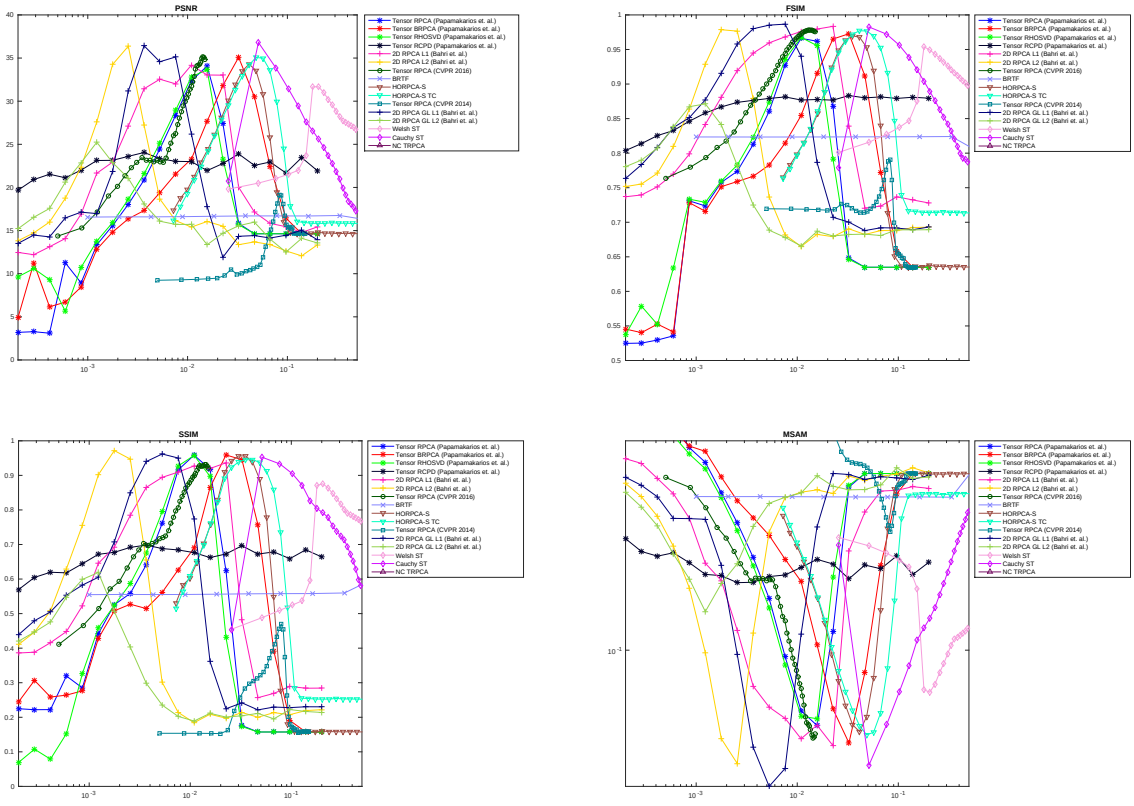

Figure C.4: Evolution of the four measures on the Yale-B benchmark at 10% salt & pepper noise

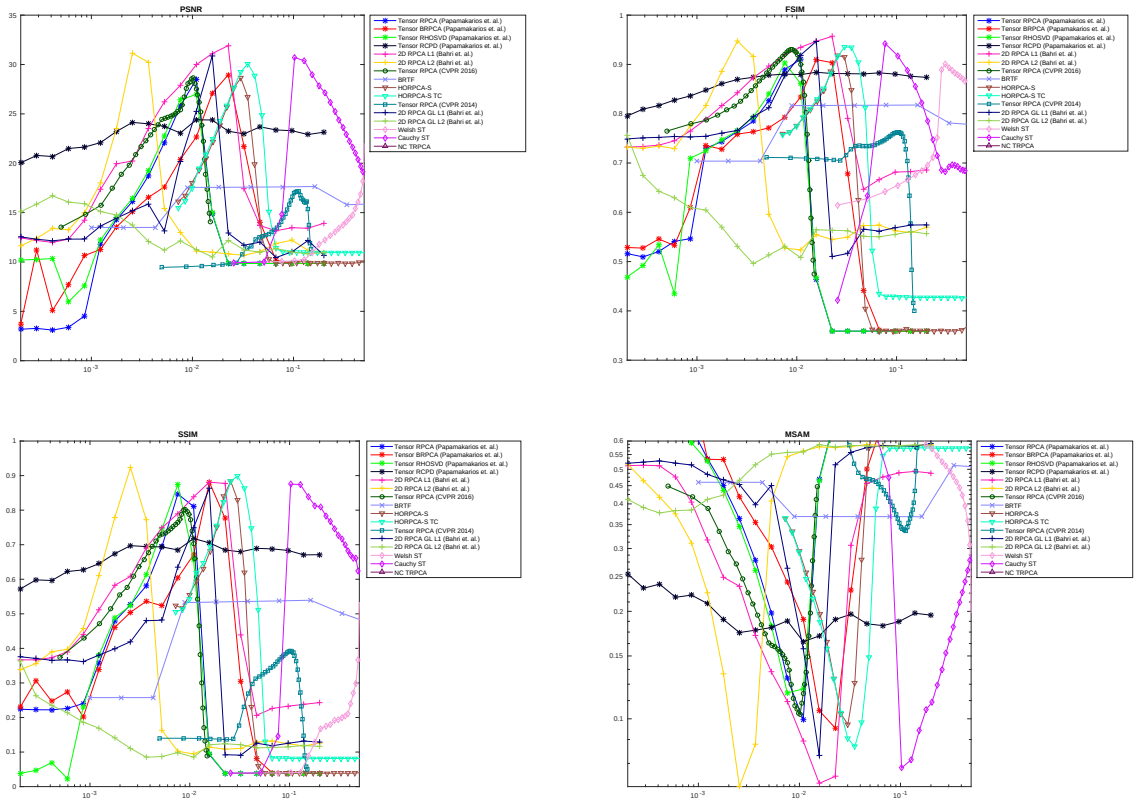

Figure C.5: Evolution of the four measures on the Yale-B benchmark at 30% salt & pepper noise

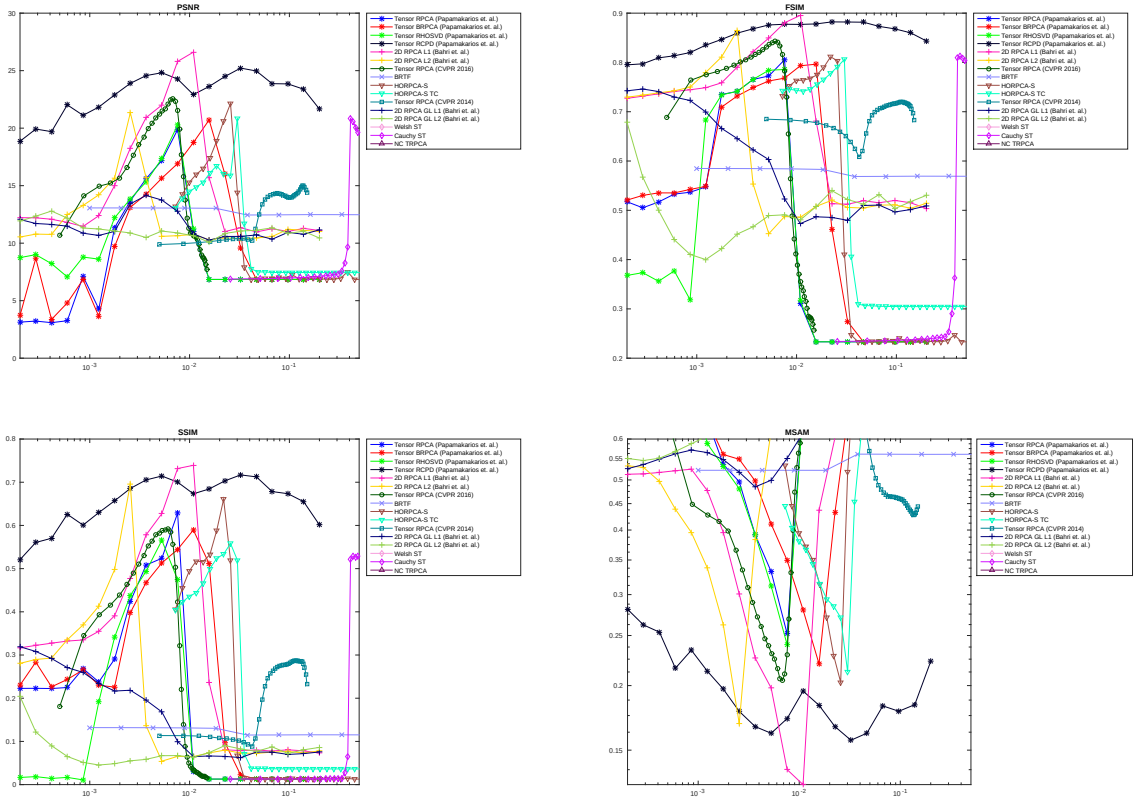

Figure C.6: Evolution of the four measures on the Yale-B benchmark at 60% salt & pepper noise

C.3 Yale-B - Patch corruption

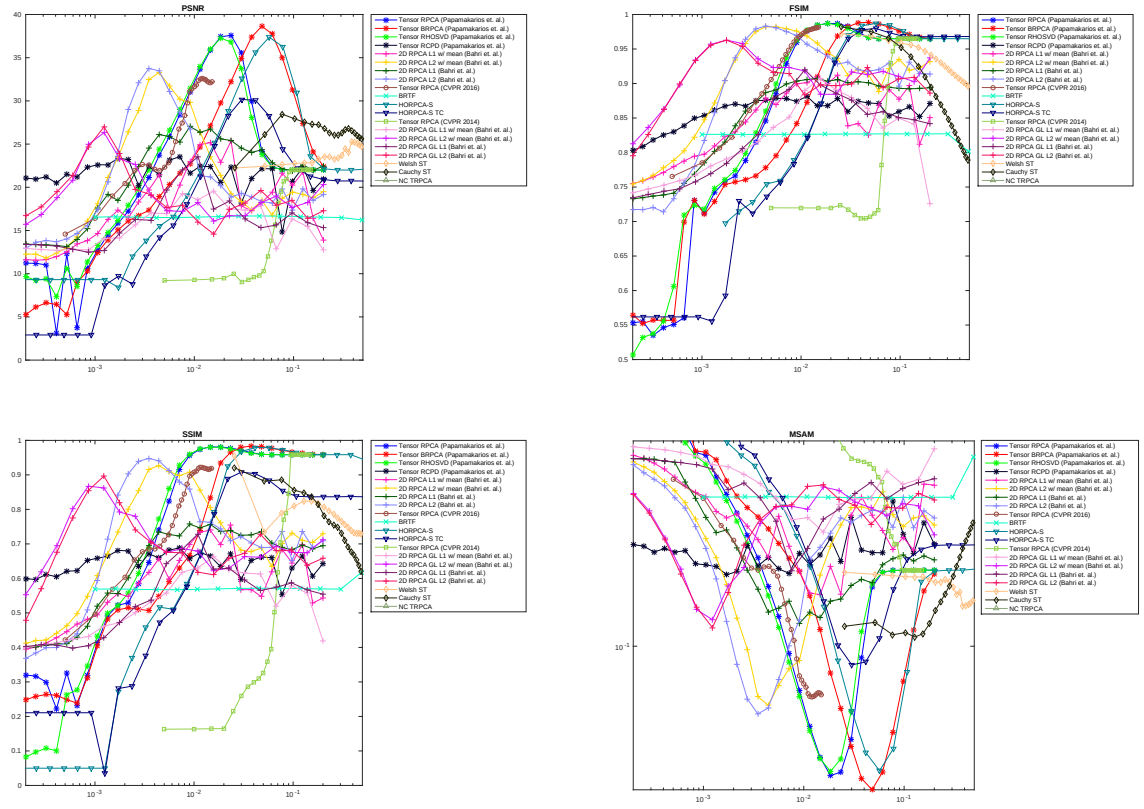

Figure C.7: Evolution of the four measures on the Yale-B Patch benchmark with max size of 50px

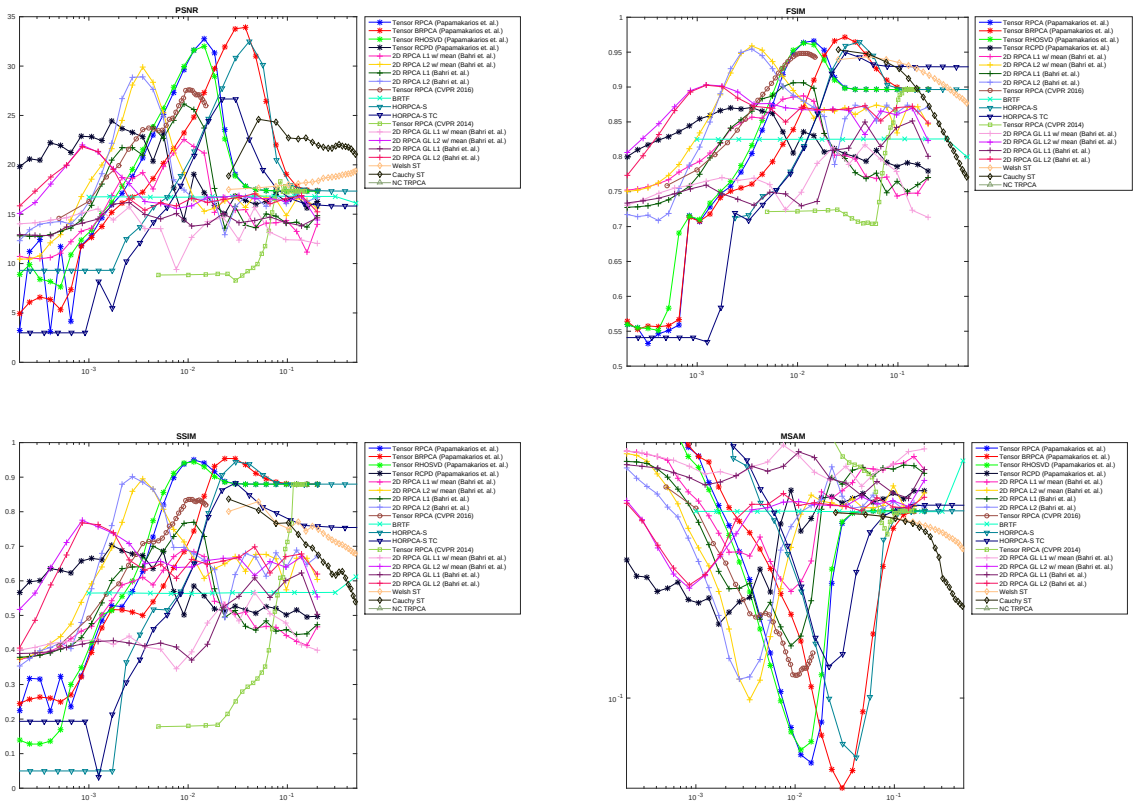

Figure C.8: Evolution of the four measures on the Yale-B Patch benchmark with max size of 100px

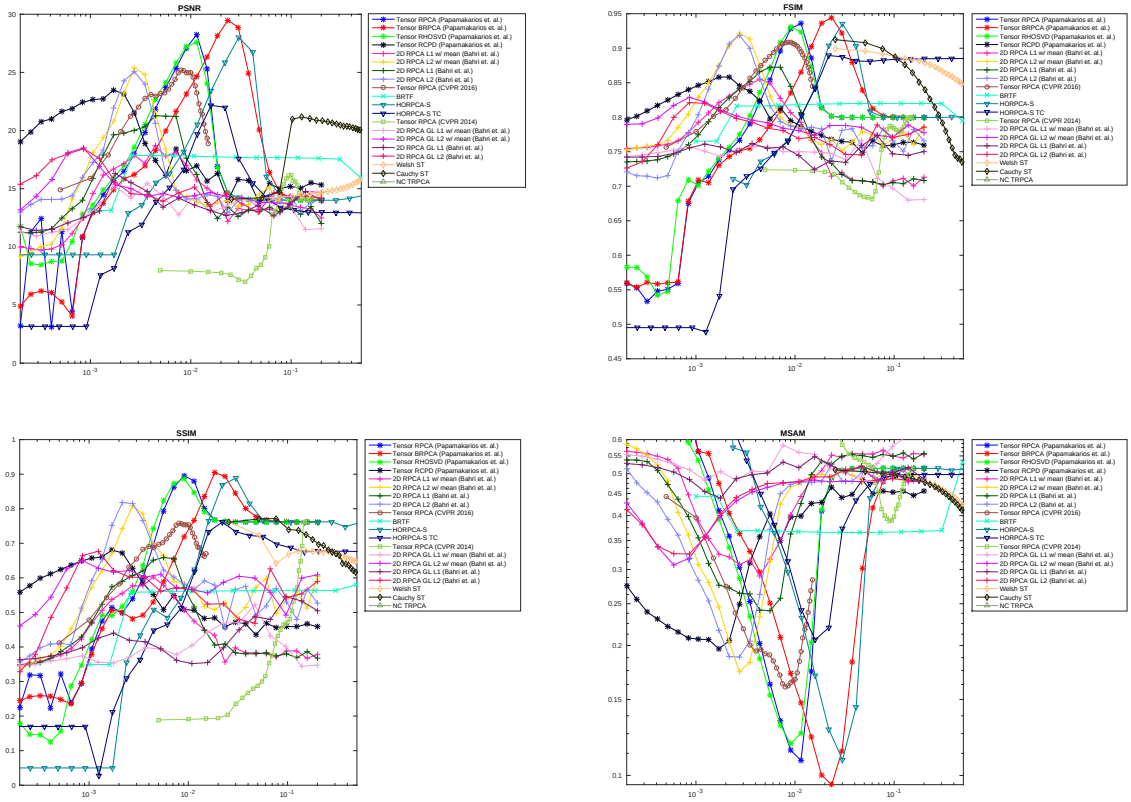

Figure C.9: Evolution of the four measures on the Yale-B Patch benchmark with max size of 160px

List of Tables

4.1 Abbreviations used for our four methods 51
4.2 Recovery performance - Frobenius penalty on the bases - simple criterion 52
4.3 Summary of our methods 64

5.1 Algorithms and their parameters 68
5.2 Algorithms and ranges of parameters 69
5.3 AUC scores - *Highway* 71
5.4 AUC scores - *Highway* with matrix methods (from Papamakarios [76]) 74
5.5 AUC scores - *Airport Hall* 76
5.6 Best performance on Yale-B sorted by FSIM - 10% Salt & Pepper noise 80
5.7 Best performance on Yale-B sorted by FSIM - 30% Salt & Pepper noise 83
5.8 Best performance on Yale-B sorted by FSIM - 60% Salt & Pepper noise 88
5.9 Reconstruction quality of the mean on Yale-B - Patch of max size 160px 89
5.10 Best performance on Yale-B sorted by PSNR - Patch of max size 50px 93
5.11 Best performance on Yale-B sorted by PSNR - Patch of max size 100px 98
5.12 Best performance on Yale-B sorted by PSNR - Patch of max size 160px 99
5.13 Best performance on Facade - 10% Salt & Pepper noise 102
5.14 Best performance on Facade - 30% Salt & Pepper noise 104
5.15 Best performance on Facade - 60% Salt & Pepper noise 106

B.1 Best parameters on Facade - 10% Salt & Pepper noise 125
B.2 Best parameters on Facade - 30% Salt & Pepper noise 126
B.3 Best parameters on Facade - 60% Salt & Pepper noise 127
B.4 Best parameters on Yale-B - 10% Salt & Pepper noise 128
B.5 Best parameters on Yale-B - 30% Salt & Pepper noise 129
B.6 Best parameters on Yale-B - 60% Salt & Pepper noise 130
B.7 Best parameters on Yale-B - Patch of max size 50px 131
B.8 Best parameters on Yale-B - Patch of max size 100px 132
B.9 Best parameters on Yale-B - Patch of max size 160px 133

List of Figures

2.1 Tensor fibres 10
2.2 Tensor slices 10
2.3 CP decomposition on a third-order tensor 12
2.4 HOSVD decomposition on a third-order tensor 13
2.5 PSNR and subjective quality 23
2.6 Representation of Reference Angle. Adapted from Rashmi et al. [81]. 24

4.1 ℓ_2 relative error - synthetic data - 30% noise 53
4.2 Structural recovery - synthetic data - 30% noise 54
4.3 ℓ_2 relative error - synthetic data - 60% noise 54
4.4 Structural recovery - synthetic data - 60% noise 55
4.5 Relative error on the sample mean at the 30% and 60% noise levels 55
4.6 Rank recovery and sparsity 56
4.7 Convergence on synthetic data 58

5.1 DET curves on the *Highway* dataset 70
5.2 Background subtraction results on the *Highway* dataset - part 1 72
5.3 Background subtraction results on the *Highway* dataset - part 2 73
5.4 DET curves on the *Airport Hall* dataset 75
5.5 Background subtraction results on the *Airport Hall* dataset - part 1 77
5.6 Background subtraction results on the *Airport Hall* dataset - part 2 78
5.7 Best results per FSIM on Yale-B with 10% S & P noise - first image 81
5.8 Best results per FSIM on Yale-B with 10% S & P noise - detail of the third image . . 82
5.9 Best results per FSIM on Yale-B with 30% S & P noise 84
5.10 Best results per FSIM on Yale-B with 30% S & P noise - detail of the third image . . 85
5.11 Best results per FSIM on Yale-B with 60% S & P noise 86
5.12 Best results per FSIM on Yale-B with 60% S & P noise - detail of the third image . . 87
5.13 Sample results on Yale-B with patch of max size 160px 89
5.14 Recovered mean - Yale-B - Patch of max size 160px 90
5.15 Best results per FSIM on Yale-B with patch of max size 50px 91
5.16 Best results per PSNR on Yale-B with patch of max size 50px 92
5.17 Best results per FSIM on Yale-B with patch of max size 100px 94
5.18 Best results per PSNR on Yale-B with patch of max size 100px 95
5.19 Best results per FSIM on Yale-B with patch of max size 160px 96
5.20 Best results per PSNR on Yale-B with patch of max size 160px 97
5.21 Original image for the Facade benchmark 100
5.22 Best results per FSIMc on Facade with 10% S & P noise 101
5.23 Best results per FSIMc on Facade with 30% S & P noise 103
5.24 Best results per FSIMc on Facade with 60% S & P noise 105

6.1 Graphical model of our Bilinear Probabilistic Robust PCA 109

C.1 Image quality measures - Facade 10% 134
C.2 Image quality measures - Facade 30% 135
C.3 Image quality measures - Facade 60% 136
C.4 Image quality measures - Yale-B 10% S & P 137
C.5 Image quality measures - Yale-B 30% S & P 138
C.6 Image quality measures - Yale-B 60% S & P 139
C.7 Image quality measures - Yale-B Patch 10% 140
C.8 Image quality measures - Yale-B Patch 30% 141
C.9 Image quality measures - Yale-B Patch 60% 142

Bibliography

[1] A. P. Dempster N. M. Laird, D. B. R. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38. pages 22, 28

[2] Absil, P. a., Mahony, R., and Sepulchre, R. (2008). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press. pages 119

[3] Amari, S. and Nagaoka, H. (2000). *Methods of Information Geometry*, volume 191 of *Translations of Mathematical monographs*. Oxford University Press. pages 21

[4] Anandkumar, A., Jain, P., Shi, Y., and Niranjan, U. N. (2016). Tensor vs Matrix Methods: Robust Tensor Decomposition under Block Sparse Perturbations. In Gretton, A. and Christian, R. C., editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, Cadiz, Spain. JMLR.org. pages 36, 66

[5] Anderson, T. W. (1946). The Non-Central Wishart Distribution and Certain Problems of Multivariate Statistics. *The Annals of Mathematical Statistics*, 17(4):409–431. pages 21

[6] Babacan, S. D., Luessi, M., Molina, R., and Katsaggelos, A. K. (2012). Sparse Bayesian Methods for Low-Rank Matrix Estimation. *IEEE Transactions on Signal Processing*, 60(8):3964–3977. pages 29, 30, 108

[7] Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2011). Optimization with Sparsity-Inducing Penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106. pages 47

[8] Bader, B. W., Kolda, T. G., and others (2015). MATLAB Tensor Toolbox Version 2.6. Available online. pages 62

[9] Bartels, R. H. and Stewart, G. W. (1972). Solution of the Matrix Equation AX + XB = C [F4]. *Commun. ACM*, 15(9):820–826. pages 43

[10] Basri, R. and Jacobs, D. (2003). Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233. pages 79

[11] Beck, A. and Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202. pages 47

[12] Benner, P., Mehrmann, V., Sima, V., Van Huffel, S., and Varga, A. (1999). SLICOTA Subroutine Library in Systems and Control Theory. In *Applied and Computational Control, Signals, and Circuits SE - 10*, pages 499–539. Birkhäuser Boston, Boston, MA. pages 43

[13] Binder, K. (2001). Ising Model. pages 30

[14] Bishop, C. (1999). Variational principal components. In *9th International Conference on Artificial Neural Networks: ICANN '99*, volume 1999, pages 509–514. IEE. pages 28

[15] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*, volume 4. Springer. pages 21, 22, 26, 28, 38

[16] Bouhamidi, A. and Jbilou, K. (2008). A note on the numerical approximate solutions for generalized Sylvester matrix equations with applications. *Applied Mathematics and Computation*, 206(2):687–694. pages 43

[17] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA. pages 17, 18, 58

[18] Bubeck, S. (2014). Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357. pages 18, 58

[19] Candes, E., Romberg, J., and Tao, T. (2005). Stable Signal Recovery from Incomplete and Inaccurate Measurements. *Science*, 40698:1–15. pages 2, 26

[20] Candès, E. J. (2008). The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592. pages 2, 26

[21] Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM*, 58(3):1–37. pages 26, 120

[22] Candes, E. J. and Tao, T. (2005). Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215. pages 2, 26

[23] Carroll, J. D. and Chang, J.-J. (1970). Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319. pages 12

[24] Chan, T. F. (1982). An Improved Algorithm for Computing the Singular Value Decomposition. *ACM Transactions on Mathematical Software*, 8(1):72–83. pages 122

[25] Chen, X., Han, Z., Wang, Y., Zhao, Q., Meng, D., and Tang, Y. (2016). Robust Tensor Factorization with Unknown Noise. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5213–5221. pages 38, 66

[26] Cheng, H., Yu, Y., Zhang, X., Xing, E., and Schuurmans, D. (2016). Scalable and Sound Low-Rank Tensor Learning. *ICML 2016*, 51. pages 37

[27] De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). A Multilinear Singular Value Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278. pages 13

[28] Derksen, H. (2016). On the Nuclear Norm and the Singular Value Decomposition of Tensors. *Foundations of Computational Mathematics*, 16(3):779–811. pages 14

[29] Ding, C. (2008). Robust Tensor Factorization Using R 1 Norm. *Proceedings of the 21st IEEE Conference on Computer Vision and Pattern Recognition - CVPR '08*, 1:1–8. pages 34

[30] Ding, C. and Ye, J. (2005). Two-Dimensional Singular Value Decomposition (2DSVD) for 2D Maps and Images. *Proc. SIAM Int'l Conf. Data Mining (SDM'05)*, pages 32–43. pages 27

[31] Ding, X., He, L., and Carin, L. (2011). Bayesian robust principal component analysis. *IEEE Transactions on Image Processing*, 20(99):11. pages 28, 29

[32] Dosselmann, R. and Dong Yang, X. (2011). A comprehensive assessment of the structural similarity index. *SIViP*, 5(1):81–91. pages 24

[33] Ehlers, R. and Brooks, S. (2003). Bayesian Analysis of Order Uncertainty in ARIMA Models. *Working Paper*, pages 1–19. pages 120

[34] Enciso-Mora, V., Neal, P., and Subba Rao, T. (2006). Efficient order selection algorithms for integer valued ARMA processes. *Journal of Time Series*, 30(1):1–18. pages 120

[35] Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874. pages 68

[36] Filipović, M. and Jukić, A. (2015). Tucker factorization with missing data with application to low- $$n$$ n -rank tensor completion. *Multidimensional Systems and Signal Processing*, 26(3):677–692. pages 33

[37] Fox, C. W. and Roberts, S. J. (2012). A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, 38(2):85–95. pages 21, 22

[38] Gandy, S., Recht, B., and Yamada, I. (2011). Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010. pages 30

[39] Gilbert, J. R., Moler, C., and Schreiber, R. (1992). Sparse Matrices in MATLAB: Design and Implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356. pages 62

[40] Goldfarb, D. and Qin, Z. T. (2014). Robust Low-Rank Tensor Recovery: Models and Algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1):225–253. pages 31, 32, 37, 53, 66, 67, 80, 90, 120

[41] Golub, G., Nash, S., and Van Loan, C. (1979). A Hessenberg-Schur method for the problem AX + XB= C. *IEEE Transactions on Automatic Control*, 24(6):909–913. pages 43, 121

[42] Goyette, N., Jodoin, P. M., Porikli, F., Konrad, J., and Ishwar, P. (2012). changedetection.net: A new change detection benchmark dataset. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. pages 69

[43] Gu, Q., Gui, H., and Han, J. (2014). Robust Tensor Decomposition with Gross Corruption. In *Advances in Neural Information Processing Systems*, pages 1422–1430. pages 31

[44] Guan, Y. and Dy, J. (2009). Sparse probabilistic principal component analysis. *International Conference on Artificial . . .* , pages 185–192. pages 108

[45] Gupta, A. K. and Nagar, D. K. (1999). *Matrix Variate Distributions*. Chapman & Hall. pages 20, 21

[46] Halko, N., Martinsson, P.-G., and Tropp, J. A. (2009). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. pages 62

[47] Harshman, R. a. (1970). Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16(10):1– 84. pages 12

[48] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA. pages 2, 38

[49] Hitchcock, F. L. (1927). The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189. pages 12, 13

[50] Hu, W., Tao, D., Zhang, W., Xie, Y., and Yang, Y. (2015). A New Low-Rank Tensor Model for Video Completion. *arXiv preprint*, pages 1–8. pages 14, 17, 36

[51] Huan, G., Li, Y., and Song, Z. (2016). A novel robust principal component analysis method for image and video processing. *Applications of Mathematics*, 61(2):197–214. pages 29

[52] Jian Yang, Zhang, D., Frangi, A., and Jing-yu Yang (2004). Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1):131–137. pages 28

[53] Jianhua Zhao, Yu, P. L. H., and Kwok, J. T. (2012). Bilinear Probabilistic Principal Component Analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):492–503. pages 20, 28, 29

[54] Kågström, B. and Poromaa, P. (1996). LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs. *ACM Transactions on Mathematical Software*, 22(1):78–103. pages 43

[55] Kolda, T. G. and Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500. pages 9, 10, 12, 13

[56] Kourouklis, S. and Moschopoulos, P. (1985). On the distribution of the trace of a non-central Wishart. *Metron*, XLIII(1-2):85–92. pages 21

[57] Kroonenberg, P. M. (1983). *Three-Mode Principal Component Analysis.* DSWO Press. pages 13

[58] Larsen, R. M. (1998). Lanczos bidiagonalization with partial reorthogonalization. *DAIMI Report Series*, 27(537). pages 62

[59] Lei Yuan, Jun Liu, and Jieping Ye (2013). Efficient Methods for Overlapping Group Lasso. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(9):2104–2116. pages 7

[60] Li, L., Huang, W., Gu, I.-H., and Tian, Q. (2004). Statistical Modeling of Complex Backgrounds for Foreground Object Detection. *IEEE Transactions on Image Processing*, 13(11):1459–1472. pages 75

[61] Li, S. Z. (2009). *Markov Random Field Modeling in Image Analysis*. Advances in Pattern Recognition. Springer London, London. pages 30

[62] Li, T., Weng, P. C.-Y., Chu, E. K.-W., and Lin, W.-W. (2013). Large-scale Stein and Lyapunov Equations, Smith Method, and Applications. *Numer. Algorithms*, 63(4):727–752. pages 43

[63] Lin, Z., Chen, M., and Ma, Y. (2010). The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *arXiv:1009.5055*, page 23. pages 26, 59

[64] Lin Zhang, Lei Zhang, Xuanqin Mou, and Zhang, D. (2011). FSIM: A Feature Similarity Index for Image Quality Assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386. pages 24

[65] Liu, Y., Shang, F., Cheng, H., Cheng, J., and Tong, H. (2014). Factor Matrix Trace Norm Minimization for Low-Rank Tensor Completion. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 866–874. Society for Industrial and Applied Mathematics, Philadelphia, PA. pages 33

[66] Lu, C., Feng, J., Chen, Y., Liu, W., Lin, Z., and Yan, S. (2016). Tensor Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Tensors via Convex Optimization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pages 10, 14, 17, 18, 36, 66, 67, 75, 102, 120

[67] Mairal, J., Jenatton, R., Obozinski, G., and Bach, F. (2010). Network Flow Algorithms for Structured Sparsity. *Advances in Neural Information Processing Systems 23*, (August):1–9. pages 7

[68] Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The DET Curve in Assessment of Detection Task Performance. *Proc. Eurospeech '97*, pages 1895–1898. pages 69

[69] Mateos, G. and Giannakis, G. B. (2012). Robust PCA as Bilinear Decomposition With Outlier-Sparsity Regularization. *IEEE Transactions on Signal Processing*, 60(10):5176–5190. pages 50

[70] Mathai, A. and Pillai, K. (1982). Further results on the trace of a noncentral wishart matrix. *Communications in Statistics - Theory and Methods*, 11(10):1077–1086. pages 21

[71] MATLAB. *version 8.5.0.197613 (R2015a)*. The MathWorks Inc., Natick, Massachusetts. pages 61

[72] Mohammadi, P. (2014). Subjective and Objective Quality Assessment of Image: A Survey. *arXiv preprint arXiv: . . .* , (June):1–50. pages 22, 24

[73] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press. pages 21, 22, 38, 63, 107

[74] Nyquist, H. (1928). Certain Topics in Telegraph Transmission Theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644. pages 2

[75] Panagakis, Y., Kotropoulos, C., and Arce, G. (2010). Non-Negative Multilinear Principal Component Analysis of Auditory Temporal Modulations for Music Genre Classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):576–588. pages 119

[76] Papamakarios, G. (2014). *Robust Low-Rank Modelling on Matrices and Tensors*. PhD thesis, Department of Computing, Imperial College London. pages 32, 34, 66, 67, 69, 70, 74, 79, 80, 90, 102, 143

[77] Papamakarios, G., Panagakis, Y., and Zafeiriou, S. (2014). Generalised Scalable Robust Principal Component Analysis. In *British Machine Vision Conference (BMVC)*, pages 1–11. pages 120

[78] Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2(6):559572. pages 1, 25

[79] Ramadan, M. A., El-Shazly, N. M., and Selim, B. I. (2010). A Hessenberg method for the numerical solutions to types of block Sylvester matrix equations. *Mathematical and Computer Modelling*, 52(9-10):1716–1727. pages 43

[80] Ramamoorthi, R. and Hanrahan, P. (2001). An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques - SIGGRAPH '01*, volume 64, pages 497–500, New York, New York, USA. ACM Press. pages 79

[81] Rashmi, S., Addamani, S., and Ravikiran, S. (2014). Spectral Angle Mapper Algorithm for Remote Sensing Image Classification. *International Journal of Innovative Science, Engineering & Technology*, 1(4):201–205. pages 24, 144

[82] Rauhut, H., Schneider, R., and Stojanac, Z. (2016). Low rank tensor recovery via iterative hard thresholding. *Proc. SampTA 2013*, (3):1–34. pages 34

[83] Recht, B., Fazel, M., and Parrilo, P. A. (2007). Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization. *Optimization Online*, 52(3):1–33. pages 8

[84] Romera-Paredes, B. and Pontil, M. (2013). A New Convex Relaxation for Tensor Completion. *arXiv preprint arXiv:1307.4653*, pages 1–17. pages 17

[85] Schäcke, K. (2013). On the Kronecker Product. pages 8

[86] Schein, A., Paisley, J., Blei, D. M., and Wallach, H. (2015). Bayesian Poisson Tensor Factorization for Inferring Multilateral Relations from Sparse Dyadic Event Counts Categories and Subject Descriptors. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009. pages 39

[87] Shang, F., Liu, Y., and Cheng, J. (2014). Generalized Higher-Order Tensor Decomposition via Parallel ADMM. *Aaai*, pages 1279–1285. pages 33

[88] Shannon, C. E. (1949). Communication in the Presence of Noise. *Proceedings of the IRE*, 37(1):10–21. pages 2

[89] Simoncini, V. (2013). Computational methods for linear matrix equations. *SIAM Review*, pages 1–68. pages 43

[90] Sorensen, D. C. and Zhou, Y. (2003). Direct methods for matrix Sylvester and Lyapunov equations. *Journal of Applied Mathematics*, 2003(6):277–303. pages 43

[91] Sun, Y., Tao, X., Li, Y., and Lu, J. (2013). Robust two-dimensional principal component analysis via alternating optimization. In *2013 IEEE International Conference on Image Processing*, pages 340–344. IEEE. pages 27

[92] Sun, Y., Tao, X., Li, Y., and Lu, J. (2015). Robust 2D Principal Component Analysis: A Structured Sparsity Regularized Approach. *IEEE Transactions on Image Processing*, 24(8):2515–2526. pages 27

[93] Sundin, M., Chatterjee, S., and Jansson, M. (2015). Bayesian Learning for Robust Principal Component Analysis. *European Conference on Signal Processing*, pages 2406–2410. pages 30

[94] Sundin, M., Chatterjee, S., Jansson, M., and Rojas, C. R. (2014). Relevance singular vector machine for low-rank matrix sensing. *2014 International Conference on Signal Processing and Communications, SPCOM 2014*. pages 30

[95] Tarantola, A. (2004). *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA. pages 2

[96] Tassa, Y. (2013). MMX - Multithreaded matrix operations on N-D matrices. Available online. pages 61

[97] Tibshirani, R. (1994). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288. pages 2

[98] Tikhonov, A. N., Goncharsky, A. V., Stepanov, V. V., and Yagola, A. G. (1995). *Numerical Methods for the Solution of Ill-Posed Problems*. Springer Netherlands, Dordrecht. pages 2

[99] Tipping, M. (2001a). Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211–244. pages 27, 30

[100] Tipping, M. E. (2001b). Sparse Kernel Principal Component Analysis. In *Advances in Neural Information Processing Systems*, pages 633–639. pages 27

[101] Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311. pages 13

[102] Vervliet, N., Debals, O., Sorber, L., Van Barel, M., and De Lathauwer, L. (2016). Tensorlab 3.0. pages 62

[103] Wang, N. and Yeung, D. Y. (2013). Bayesian robust matrix factorization for image and video processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1785–1792. IEEE. pages 29

[104] Wang, Z. and Bovik, A. C. (2003). Multi-Scale Structural Similarity For Image Quality Assessment. *Proceedings of the IEEE Asilomar Conference on Signals*, 2(Ki L):1398–1402. pages 24

[105] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612. pages 23

[106] Wipf, D. and Nagarajan, S. (2008). A New View of Automatic Relevance Determination. *Compute*, 20(2):16251632. pages 27

[107] Wipf, D., Palmer, J., and Rao, B. (2004). Perspectives on Sparse Bayesian Learning. *Computer Engineering*, 16(1):249. pages 27

[108] Xiong, L., Chen, X., Huang, T.-k., Schneider, J., and Carbonell, J. G. (2010). Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. *Proceedings of the SIAM International Conference on Data Mining*, pages 211–222. pages 120

[109] Yang, Y., Feng, Y., and Suykens, J. A. K. (2015a). A Rank-One Tensor Updating Algorithm for Tensor Completion. *IEEE Signal Processing Letters*, 22(10):1633–1637. pages 37, 38

[110] Yang, Y., Feng, Y., and Suykens, J. A. K. (2015b). Robust Low-Rank Tensor Recovery With Regularized Redescending M-Estimator. *IEEE Transactions on Neural Networks and Learning Systems*, 27(9):1933–1946. pages 32, 49, 66, 67, 120

[111] Ye, J. (2005). Generalized Low Rank Approximations of Matrices. *Machine Learning*, 61(1-3):167–191. pages 28

[112] Yu, S. and Bi, J. (2008). Probabilistic Interpretations and Extensions for a Family of 2D PCA-style Algorithms. In *Proc. KDD Workshop Data Min. Using Matri. Tensors*, number Section 4, pages 1–7. pages 28

[113] Zhang, S., Qian, H., and Gong, X. (2010). An Alternating Proximal Splitting Method with Global Convergence for Nonconvex Structured Sparsity Optimization. pages 2330–2336. pages 7

[114] Zhang, X. and Wang, D. (2013). Simultaneous Rectification and Alignment via Robust Recovery of Low-rank Tensors. In *Nips*, pages 1–9. pages 35

[115] Zhang, Z., Ely, G., Aeron, S., Hao, N., and Kilmer, M. (2014). Novel methods for multilinear data completion and de-noising based on tensor-SVD. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3842–3849. pages 14, 17, 18, 35, 36, 66, 67, 102

[116] Zhao, Q., Meng, D., Xu, Z., Zuo, W., and Zhang, L. (2014). Robust Principal Component Analysis with Complex Noise. *Proceedings of The 31st International Conference on Machine Learning*, pages 55–63. pages 29, 38

[117] Zhao, Q., Zhang, L., and Cichocki, A. (2015). Bayesian CP Factorization of Incomplete Tensors with Automatic Rank Determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1–15. pages 38

[118] Zhao, Q., Zhou, G., Zhang, L., Cichocki, A., and Amari, S.-I. (2016). Bayesian Robust Tensor Factorization for Incomplete Multiway Data. *IEEE Transactions on Neural Networks and Learning Systems*, 27(4):736–748. pages 38, 66, 75

[119] Zhou, J., Bhattacharya, A., Herring, A. H., and Dunson, D. B. (2014). Bayesian factorizations of big sparse tensors. *Journal of the American Statistical Association*, 1459(1994):00–00. pages 38

[120] Zhou, X., Yang, C., and Yu, W. (2013). Moving object detection by detecting contiguous outliers in the low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(3):597–610. pages 29

[121] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 67:768. pages 120